# ASReview Software Documentation

***Release 1.5+21.gdc68354***

**ASReview LAB developers, Utrecht University**

**Feb 01, 2024**

# CONTENTS

Welcome to the ASReview LAB Documentation!

CHAPTER

ONE

# GET STARTED

## 1.1 What is ASReview LAB?

ASReview LAB is a free (Libre) open-source machine learning tool for screening and systematically labeling a large collection of textual data. It's sometimes referred to as a tool for title and abstract screening in systematic reviews or meta-analyses, but it can handle any type of textual data that must be screened systematically, see the paper published in Nature Machine Intelligence.

ASReview LAB implements three different options:

- **Oracle:** Screen textual data in interaction with the active learning model. The reviewer is the 'oracle', making the labeling decisions.
- **Simulation:** Evaluate the performance of active learning models on fully labeled data.
- **Validation:** Validate labels provided by another screener or derived from an LLM or AI, and explore benchmark datasets without being an oracle.

ASReview LAB is one of the products of the ASReview research project initiated at Utrecht University, which has grown into a vivid community of researchers, users, and developers from around the world.

https://youtu.be/k-a2SCq-LtA

## 1.2 What is active learning?

Artificial Intelligence (AI) and machine learning has allowed the development of AI-aided pipelines that assist in finding relevant texts for search tasks. A well-established approach to increasing the efficiency of screening large amounts of textual data is screening prioritization through Active Learning: a constant interaction between a human who labels records and a machine learning model which selects the most likely relevant record based on a minimum training dataset. The active learning cycle is repeated until the annotator is sufficiently confident they have seen all relevant records. Thus, the machine learning model is responsible for ranking the records and the human provides the labels, this is called Researcher-In-The-Loop (RITL).

It allows the screening of large amounts of text in an intelligent and time-efficient manner. ASReview LAB, published in Nature Machine Intelligence, has shown the benefits of active learning, reducing up to 95% of the required screening time.

## 1.3 Labeling workflow with ASReview

Start and finish a systematic labeling process with ASReview LAB by following these steps:

1. Create a dataset with potentially relevant records you want to screen systematically. Improve the quality of the data and specify clear reviewing (inclusion/exclusion) criteria
2. Specify a stopping criterion
3. *Start ASReview LAB*
4. *Create a project*
5. *Import your dataset*
6. *Select Prior Knowledge*
7. Select the four components of the *Active learning model* (feature extractor, classifier, balancing method, query strategy)
8. Wait until the warm up of the AI is ready (the software is extracting the features and trains the classifier on the prior knowledge)
9. Start *Screening* until you reach your stopping criterion
10. At any time, you can export the *dataset* the labeling decisions or the entire *project*.

## 1.4 Quick start

1. Check if Python 3.8 or later is installed (if not, install Python)

```
python --version
```

2. Install ASReview LAB

```
pip install asreview
```

3. Open ASReview LAB

```
asreview lab
```

4. Click *Create* to create a project
5. Select a mode (Oracle, Validation, Simulation)
6. Name the project, and if you want, add an author name(s) and type a description
7. Import a dataset you want to review, or select a benchmark dataset (only available for the Validation and Simulation mode)
8. Add prior knowledge. Select at least 1 relevant and 1 irrelevant record to warm up the AI. You can search for a specific record or request random records
9. Select the four components of the active learning model, or rely on the default settings that have shown fast and excellent performance in many simulation studies
10. ASReview LAB starts extracting the features and runs the classifier with the prior knowledge

You're ready to start labeling your data! All your labeling actions are automatically saved, so there is no need to click the save button (we don't even have one).

## 1.5 ASReview LAB terminology

When you do text screening for a systematic review in ASReview LAB, it can be useful to know some basic concepts about systematic reviewing and machine learning to understand. The following overview describes some terms you might encounter as you use ASReview LAB.

**Active learning model**
An active learning model is the combination of four elements: a feature extraction technique, a classifier, a balance, and a query strategy.

**ASReview**
ASReview stands for *Active learning for Systematic Reviews* or *AI-assisted Systematic Reviews*, depending on context. Avoid this explanation, only use as tagline.

**ASReview CLI**
ASReview CLI is the command line interface that is developed for advanced options or for running simulation studies.

**Data**
Data includes *dataset*, prior knowledge, labels, and *notes*.

**Dataset**
A dataset is the collection of *records* that the *user imports* and *exports*.

**ELAS**
ELAS stands for "Electronic Learning Assistant". It is the name of *ASReview* mascot. It is used for storytelling and to increase explainability.

**Export**
Export is the action of exporting a *dataset* or a *project* from ASReview LAB.

**Extension**
An extension is the additional element to the ASReview LAB, such as the ASReview Datatools extension.

**Import**
Import is the action of importing a *dataset* or a *project* into ASReview LAB.

**Model configuration**
Model configuration is the action of the *user* to configure the *active learning model*.

**Note**
A note is the information added by the *user* in the note field and stored in the *project file*. It can be edited on the History page.

**Project**
A project is a project created in ASReview LAB.

**Projects dashboard**
The project dashboard is the landing page containing an overview of all *projects* in ASReview LAB.

**Project file**
The project file is the `.asreview` file containing the *data* and *model configuration*. The file is *exported* from ASReview LAB and can be *imported* back.

**Project mode**
The project mode includes oracle, simulation, and validation in ASReview LAB:

**Oracle** mode is used when a *user* reviews a *dataset* systematically with interactive artificial intelligence (AI).

**Validation** mode is used when a user validates existing labels or engages in a review process without being an oracle

**Simulation** mode is used when a user simulates a review on a completely labeled dataset to see the performance of ASReview LAB.

**Status**
The project status is the stage that a *project* is at in ASReview LAB.

**Setup** refers to the fact that the *user* adds project information, *imports* the *dataset*, selects the prior knowledge, *configures the model* and initiates the first iteration of *model* training.

**In Review** refers to the fact that in oracle or validation mode, the user adds labels to *records*, or in simulation mode, the simulation is running.

**Finished** refers to the fact that in oracle or validation mode, the user decides to complete the *reviewing* process or has labeled all the records, or in simulation mode, the simulation has been completed.

**Published** refers to the fact that the user publishes the dataset and *project file* in a repository, preferably with a Digital Object Identifier (DOI).

**Record**
A record is the data point that needs to be labeled. A record can contain both information that is used for training the *active learning model*, and information that is not used for this purpose.

In the case of systematic reviewing, a record is meta-data for a scientific publication. Here, the information that is used for training purposes is the text in the title and abstract of the publication. The information that is not used for training typically consists of other metadata, for example, the authors, journal, or DOI of the publication.

**Reviewing**
Reviewing is the decision-making process on the relevance of *records* ("irrelevant" or "relevant"). It is interchangeable with Labeling, Screening, and Classifying.

**User**
The human annotator is the person who labels *records*.

**Screener**
Replacement term when the context is PRISMA-based reviewing.

## 1.6 Key principles

The use of ASReview LAB comes with five fundamental principles:

1. Humans are the oracle;
2. Code is open & results are transparent;
3. Decisions are unbiased;
4. The interface shows an AI is at work;
5. Users are responsible for importing high quality data.

## 1.7 Privacy

The ASReview LAB software doesn’t collect any information about the usage or its user. Great, isn’t it!

CHAPTER

TWO

# PRODUCTS

The following products are available:

## 2.1 ASReview LAB

ASReview LAB is an open-source and free-to-use software product resulting from an academic collaboration. It enables AI-aided systematic screening of textual data, such as in systematic reviews or meta-analyses. It runs in a web-browser and is flexible in using many different feature extractors and classifiers. For more information about the underlying infrastructure, refer to the Nature Machine Intelligence publication.

## 2.2 ASReview LAB Server

*ASReview LAB Server* expands upon the capabilities of ASReview LAB by offering additional features like authentication and collaborative model refinement. It is a self-hosted solution that ensures privacy and is suited for teams requiring secure access and collaboration.

## 2.3 Datasets

ASReview also provides access to a synergistic collection of datasets, which can be utilized for various research purposes. Explore the Synergy dataset at our GitHub datasets repository.

## 2.4 Extensions

ASReview supports for extending the software with new models, subcommands, and datasets. They can extend the functionality of ASReview LAB, and the *Command Line Interface*. There are *officially supported extensions* and community maintained extensions.

Looking to develop your own extension? See *Extensions* for detailed instructions.

### 2.4.1 Installation

Most extensions are installable from PyPI (the same way ASReview LAB is installed) or GitHub. It is preferred to follow the installation instructions provided by the extension.

The following example shows the installation of ASReview Insights, an extension for plotting and computing metrics for simulations in ASReview.

```
pip install asreview-insights
```

Extension (only) published on Github can be installed directly from the repository. Replace *{USER_NAME}* and *{REPO_NAME}* by the corresponding values of the extension.

```
pip install git@github.com:{USER_NAME}/{REPO_NAME}.git
```

### 2.4.2 Supported Extensions

The following extensions are officially supported and maintained by the maintainers of ASReview LAB. They are extensively tested and integrate well with ASReview LAB.

- **ASReview Datatools**
  - ASReview-datatools: Tool for describing, cleaning (input) data, and converting file formats via the command line.
- **ASReview Insights**
  - ASReview-insights: Advanced insights to ASReview simulations like performance plots and metrics.
- **ASReview Wordcloud**
  - ASReview-wordcloud: Create wordclouds to visualize the contents of datasets.
- **ASReview Makita**
  - ASReview-makita: ASReviews' Makita (MAKe IT Automatic) is a workflow generator for simulation studies using the command line interface of ASReview LAB. Makita can be used to simplify your own research by enabling you to effortlessly generate the framework and code for your simulation study.

### 2.4.3 Community maintained extensions

The List of extensions for ASReview LAB on the Discussion platform gives an overview of known extensions to ASReview LAB and other useful tools in the AI-aided systematic review pipeline.

CHAPTER

THREE

# RESEARCH

The open source ASReview LAB software is one of the products of the ASReview research project. The ASReview research project is a fundamental and applied research project studying the application of AI in the field of systematically reviewing large amounts of text data.

**Note:** The ASReview project is developed by researchers for researchers, and anyone is welcome to join the community!

There are still 1001 scientific papers that can be published using the ASReview products. We welcome researchers worldwide to work on papers like applying the existing models to new types of datasets (different scientific fields, other languages, multilanguage data, text data outside academia, large datasets, etcetera), adding new models and testing the performance on the available benchmark datasets, adding and testing new stopping rules or performance metrics, and so on!

## 3.1 Scientific principles

The research team works according to the Open Science principles and invests in an inclusive community contributing to the project. In short, research is conducted according to the following fundamental principles:

- Research output should be FAIR (Findable Accessible Interoperable and Reusable).
- Research should be conducted with integrity, and we commit ourselves to the Netherlands Code of Conduct for Research Integrity.
- Output should be rewarded according to the Declaration on Research Assessment (DORA).

Utrecht University has established specific regulations governing conduct for its employees. These are based on the key principles of professional and quality academic conduct and ethically-responsible research. Members of the team employed by Utrecht University, commit themselves to these regulations in all their conduct, including all work related to ASReview. Adherence to similar key principles is expected of all researchers involved in all facets of the ASReview project.

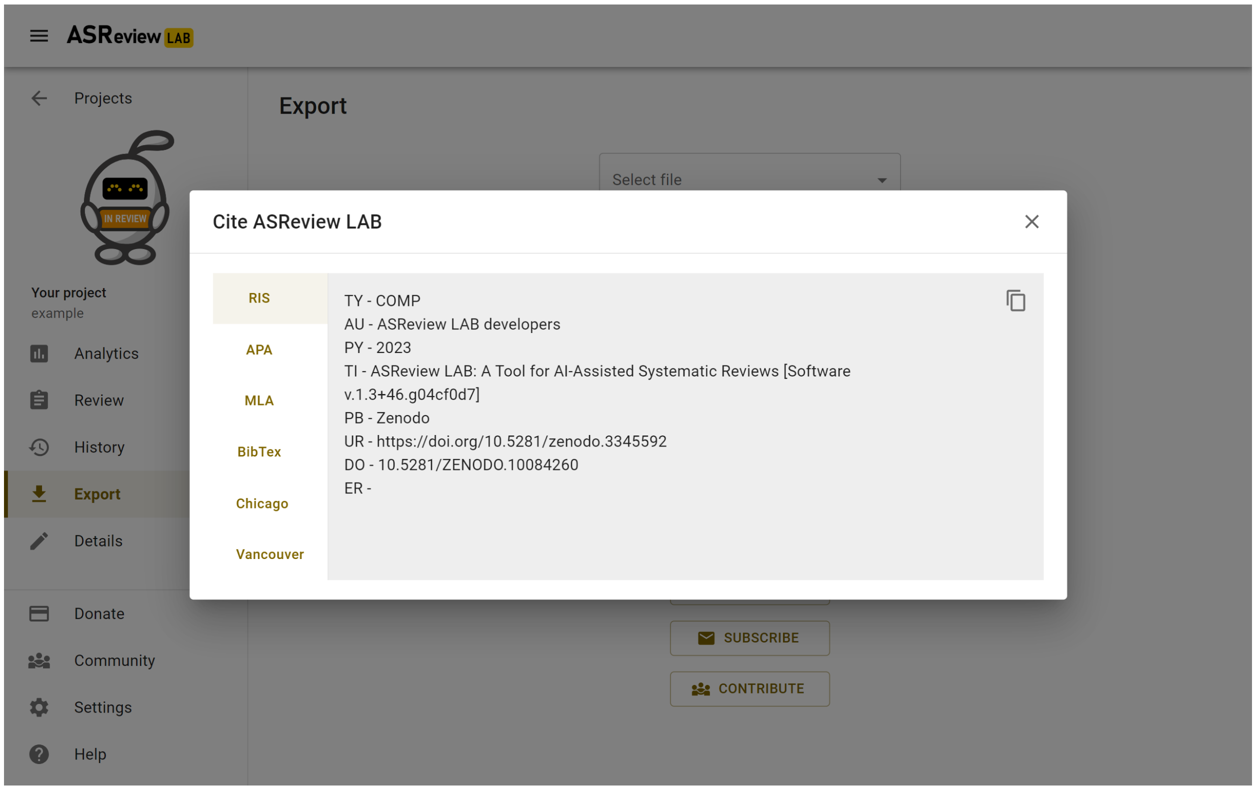
## 3.2 Cite

For scientific use, we encourage users to cite:

- The paper published in Nature Machine Intelligence to cite the **ASReview project**.
- For a detailed description of the the data model, see the paper Reproducibility and Data Storage Checklist.
- More studies related to the project can be found on asreview.ai/research.

For citing the documentation (or to download the pdf) go to Zenodo.

For citing the software **ASReview LAB**, refer to the specific release of the software, available on the export screen.



```
TY - COMP
AU - ASReview LAB developers
PY - 2023
TI - ASReview LAB: A Tool for AI-Assisted Systematic Reviews [Software
v.1.3+46.g04cf0d7]
PB - Zenodo
UR - https://doi.org/10.5281/zenodo.3345592
DO - 10.5281/ZENODO.10084260
ER -
```

CHAPTER

FOUR

# SUPPORT

Questions can be asked on GitHub Discussions. For bug reports and feature requests, please submit an issue on GitHub.

## 4.1 Donate

The ASReview software is Free and Open Source Software (FOSS). To support the development, you can donate on the ASReview donation page. Even small donations are highly appreciated!

## 4.2 Collaborate

If you are interested in (scientific) collaboration, contact Prof. Dr. Rens van de Schoot or send an email to asreview@uu.nl.

## 4.3 Contribute

How do you go from user to contributor? There are many ways to join in, and it might be less complicated than you expect. In a blogpost, we list some easy examples for first-time contributors, for example sharing your experiences or answering user questions on the Discussion platform.

Specific instructions for code-contributing are available on Github as well as instructions for developers.

**Note:** All contributions, small or large, are very much appreciated!

Questions can be asked on GitHub Discussions. For bug reports and feature requests, please submit an issue on GitHub.

CHAPTER

FIVE

# INSTALLATION

## 5.1 Install ASReview

ASReview software requires an installation of Python 3.8 or later. Detailed step-by-step instructions to install Python (and ASReview) are available for Windows and macOS/Linux users.

Install the ASReview software with Pip by running the following command in the *CMD.exe* (Windows) or *Terminal* (MacOS/Linux):

```
pip install asreview
```

Start the application with the following command (in CMD.exe or Terminal):

```
asreview lab
```

The ASReview LAB software starts in the browser. For more options on starting ASReview LAB, see *Start ASReview LAB*.

**Note:** See *Troubleshooting* for common problems during installation.

**Tip:** For users with Apple M1 computers, if you experience problems, follow the instructions.

## 5.2 Upgrade ASReview

Upgrade ASReview software with

```
pip install --upgrade asreview
```

## 5.3 Uninstall ASReview

Remove ASReview with

```
pip uninstall asreview
```

Enter `y` to confirm.

**Warning:** Note that your project files will **not** delete with this action. You find them in the *.asreview* folder in your home folder.

## 5.4 Server Installation

It is possible to run the ASReview software on a server or custom domain. Use the flags *ip* and *port* for configuration. ASReview should only be used in closed networks.

```
asreview lab --port 5555 --ip xxx.x.x.xx
```

**Warning:** Don't use the development server in production. Read the Flask documentation about deploying a Flask app to production.

## 5.5 Install with Docker

ASReview is also available as a Docker container. Make sure you have Docker installed on your machine.

To install and start ASReview LAB at http://localhost:5000, run the following:

```
docker run -p 5000:5000 ghcr.io/asreview/asreview:latest lab
```

More advanced command line options can be given afterward, like this:

```
docker run -p 9000:9000 ghcr.io/asreview/asreview lab --port 9000
```

**Tip:** ASReview LAB is now installed. Open the URL in your host web browser: `http://localhost:5000` and get started.

### 5.5.1 Mount local volume

To mount the container to your local project folder (or any other local folder), the *-v* flag can be used. To do so, adjust path-to-your-folder to your local folder. When a project folder is specified, ASReview LAB will store and load all its projects from this folder. Note that multiple containers can access the same folder.

```
docker run -p 5000:5000 -v path-to-your-folder:/project_folder ghcr.io/asreview/asreview␣
↪lab
```

### 5.5.2 Named container

To make the usage easier, you can create a named container like the following:

```
docker create --name asreview-lab -p 5000:5000 -v path-to-your-folder:/project_folder␣
↪ghcr.io/asreview/asreview lab
```

To start asreview, enter:

```
docker start asreview
```

To stop it, just use *stop* instead of *start*. You can also check which images are running with *docker ps*.

### 5.5.3 Customize the image

If you want to add more extensions, or build the Docker image yourself, check the file *Dockerfile <https://github.com/ghcr.io/asreview/asreview/tree/master/Dockerfiles>*. Modify it as you see fit, and then build and run the image with:

```
docker build -t asreview/asreview:custom .
docker run -p 5000:5000 ghcr.io/asreview/asreview:custom lab
```

# CHAPTER SIX

# START ASREVIEW LAB

After you install ASReview LAB, start the program via the command line to start using it.

```
asreview lab
```

When you are using Windows, open *CMD.exe* and run the command. When you use MacOS or Linux, you can open *Terminal* and run the command.

The information in the sections below is more advanced and not needed for the majority of the ASReview LAB users.

## 6.1 Command line arguments for starting ASReview LAB

ASReview LAB provides a powerful command line interface for running ASReview LAB with other options or even run tasks like simulations. For a list of available commands in ASReview LAB, type `asreview lab --help`.

**`asreview lab`** launches the ASReview LAB software (the frontend).

```
asreview lab [options]
```

**`-h, --help`**
Show help message and exit.

**`--ip`** `IP`
The IP address the server will listen on.

**`--port`** `PORT`
The port the server will listen on.

**`--port-retries`** `NUMBER_RETRIES`
The number of additional ports to try if the specified port is not available.

**`--enable-auth`** `ENABLE_AUTH`
Enable authentication.

**`--secret-key`** `SECRET_KEY`
Secret key for authentication.

**`--salt`** `SALT`
When using authentication, a salt code is needed for hasing passwords.

**`--flask-configfile`** `FLASK_CONFIGFILE`
Full path to a JSON file containing Flask parameters for authentication.

**--no-browser** NO_BROWSER

Do not open ASReview LAB in a browser after startup.

**--certfile** CERTFILE_FULL_PATH

The full path to an SSL/TLS certificate file.

**--keyfile** KEYFILE_FULL_PATH

The full path to a private key file for usage with SSL/TLS.

**--embedding** EMBEDDING_FP

File path of embedding matrix. Required for LSTM models.

**--clean-project** CLEAN_PROJECT

Safe cleanup of temporary files in project.

**--clean-all-projects** CLEAN_ALL_PROJECTS

Safe cleanup of temporary files in all projects.

**--seed** SEED

Seed for the model (classifiers, balance strategies, feature extraction techniques, and query strategies). Use an integer between 0 and 2^32 - 1.

## 6.2 Set environment variables

The following environment variables are available.

**ASREVIEW_PATH**

The path to the folder with project. Default *~/.asreview*.

How you set environment variables depends on the operating system and the environment in which you deploy ASReview LAB.

In MacOS or Linux operating systems, you can set environment variables from the command line. For example:

```
export ASREVIEW_PATH=~/.asreview
```

On Windows, you can use the following syntax:

```
set ASREVIEW_PATH=~/.asreview
```

To check if you set an environment variable successfully, run the following on *nix operating systems:

```
echo $ASREVIEW_PATH
```

Or the following on Windows operating systems:

```
echo %ASREVIEW_PATH%
```

## 6.3 Run ASReview LAB on localhost with a different port

By default, ASReview LAB runs on port 5000. If that port is already in use or if you want to specify a different port, start ASReview LAB with the following command:

```
asreview lab --port <port>
```

For example, start ASReview LAB on port 5001:

```
asreview lab --port 5001
```

CHAPTER

SEVEN

# TROUBLESHOOTING

ASReview LAB is advanced machine learning software. In some situations, you might run into unexpected behavior. See below for solutions to problems.

## 7.1 Unknown Command “pip”

The command line returns one of the following messages:

```
-bash: pip: No such file or directory
```

```
'pip' is not recognized as an internal or external command, operable program or batch␣
↪file.
```

First, check if Python is installed by using the following command:

```
python --version
```

If this doesn’t return a version number, then Python is either not installed or not correctly installed

Most likely, the environment variables aren’t configured correctly. Follow the step-by-step installation instruction on the ASReview website (Windows and MacOS).

However, there is a simple way to deal with correct environment variables by adding *python -m* in front of the command. For example:

```
python -m pip install asreview
```

## 7.2 Unknown command “asreview”

In some situations, the entry point “asreview” can not be found after installation. First check whether the package is correctly installed. Do this with the command *python -m asreview -h*. If this shows a decryption of the program, use *python -m* in front of all your commands. For example:

```
python -m asreview lab
```

## 7.3 Build dependencies error

The command line returns the following message:

```
"Installing build dependencies ... error"
```

This error typically happens when the version of your Python installation has been released very recently. Because of this, the dependencies of ASReview are not compatible with your Python installation yet. It is advised to install the second most recent version of Python instead. Detailed step-by-step instructions to install Python (and ASReview) are available for Windows and MacOS users.

## 7.4 Remove temporary files

In case ASReview runs into unexpected errors or doesn't work as expected, it is advised to try to remove temporary files from the project first. These files can be found in the `.asreview/` folder in your home directory. However, the easiest way to remove these files is with:

```
asreview lab --clean-all-projects
```

This will safely remove temporary files, nothing will harm your review. To clean a specific project, use

```
asreview lab --clean-project my-project
```

in which `my_project` is your project name.

CHAPTER

EIGHT

# PREPARE YOUR DATA

ASReview LAB requires a dataset containing a set of textual records (e.g., titles and abstracts of scientific papers, newspaper articles, or policy reports) obtained via a systematic search. The goal is to review all records systematically using predetermined inclusion and exclusion criteria. Also, it should be expected that only a fraction of the records in the dataset is relevant.

Datasets can be unlabeled as well as *Partially labeled data* and *Fully labeled data*. See *Project modes* for more information.

The easiest way to obtain a dataset is via a search engine or with the help of a reference manager. See *Compatibility* for reference managers export formats supported by ASReview. For more information about the format of the dataset, see *Data format*.

## 8.1 High-quality data

The algorithms of ASReview LAB work best with high-quality datasets. A high-quality dataset is a dataset with duplicate records removed, and the data is complete. Complete data implies that titles and abstracts are available for all (or most) records. See the ASReview blog Importance of Abstracts for more ideas on composing a high-quality dataset.

## 8.2 Compatibility

### 8.2.1 Citation Managers

The following table provides an overview of export files from citation managers which are accepted by ASReview.

| | .ris | .csv | .xlsx |
|---|---|---|---|
| **EndNote** | | N/A | N/A |
| **Excel** | N/A | | |
| **Mendeley** | | N/A | N/A |
| **Refworks** | | N/A | N/A |
| **Zotero** | | | N/A |

- = The data can be exported from the citation manager and imported in ASReview.
- N/A = This format does not exist.

RIS files used for screening in ASReview LAB can be imported back into the reference software and the decision labels can be found in the notes field. For more information see this instruction video.

Note: the RIS-pipeline is extensively tested for reference managers Zotero and EndNote. However, it might also work for other reference managers but is currently not supported.

**Note:** When using EndNote use the following steps to export a RIS file (.ris):

- In EndNote, click on the style selection dropdown menu from the main EndNote toolbar.
- Click “Select Another Style”.
- Browse to RefMan (RIS) Export and click “Choose”.
- Click on the file menu and select “Export”.
- Pick a name and location for the text file.
- Choose the output format RefMan (RIS) Export and click “Save”.

### 8.2.2 Search Engines

When using search engines, it is often possible to store the articles of interest in a list or folder within the search engine itself. Thereafter, you can choose from different ways to export the list/folder. When you have the option to select parts of the citation to be exported, choose the option which will provide the most information.

The export files of the following search engines have been tested for their acceptance in ASReview:

| | .ris | .tsv | .csv | .xlsx |
|---|---|---|---|---|
| **CINAHL (EBSCO)** | | N/A | X | N/A |
| **Cochrane** | | N/A | | N/A |
| **Embase** | | N/A | | |
| **Eric (Ovid)** | * | N/A | N/A | N/A |
| **Psychinfo (Ovid)** | * | N/A | N/A | N/A |
| **Pubmed** | X | N/A | X | N/A |
| **Scopus** | | N/A | | N/A |
| **Web of Science** | | N/A | N/A | N/A |

- = The data can be exported from the search engine and imported in ASReview.
- N/A = This format does not exist.
- X = Not supported, (see *Data format* for other options).

* Make sure to uncheck all inclusion options (e.g., “URL”) when exporting from Ovid.

**Tip:** If the export of your search engine is not accepted in ASReview, you can also try the following: import the search engine file first into one of the citation managers mentioned in the previous part, and export it again into a format that is accepted by ASReview.

### 8.2.3 Systematic Review Software

There are several software packages available for systematic reviewing, see https://www.nature.com/articles/s42256-020-00287-7. Some of them use machine learning, while other focus on screening and management. The overview below shows an overview of alternative software programs and the compatibility with ASReview.

| | .ris | .tsv | .csv | .xlsx |
|---|---|---|---|---|
| **Abstrackr** | | N/A | | N/A |
| **Covidence*** | | N/A | | N/A |
| **Distiller** | X | N/A | ** | ** |
| **EPPI-reviewer** | | N/A | N/A | X |
| **Rayyan** | | N/A | | N/A |
| **Robotreviewer** | N/A | N/A | N/A | N/A |

- = The data can be exported from the third-party review software and imported in ASReview.
- N/A = This format does not exist.
- X = Not supported.

* When using Covidence it is possible to export articles in `.ris` format for different citation managers, such as EndNote, Mendeley, Refworks and Zotero. All of these are compatible with ASReview.

** When exporting from Distiller and if the following error occurs `Unable to parse string "Yes (include)" at position 0` set the `sort references by` to `Authors`. Then the data can be imported in ASReview.

CHAPTER

NINE

# DATA FORMAT

To carry out a systematic review with ASReview on your own dataset, your data file needs to adhere to a certain format. ASReview accepts the following formats:

## 9.1 Tabular file format

Tabular datasets with extensions `.csv`, `.tab`, `.tsv`, or `.xlsx` can be used in ASReview LAB. CSV and TAB files are preferably comma, semicolon, or tab-delimited. The preferred file encoding is *UTF-8* or *latin1*.

For tabular data files, the software accepts a set of predetermined column names:

Table 1: Table with column name definitions

| Name | Column names | Mandatory |
|---|---|---|
| Title | title, primary_title | yes* |
| Abstract | abstract, abstract note | yes* |
| Keywords | keywords | no |
| Authors | authors, author names, first_authors | no |
| DOI | doi | no |
| URL | url | no |
| Included | final_included, label, label_included, included_label, included_final, included, included_flag, include | no |

* Only a title or an abstract is mandatory.

**Title, Abstract** Each record (i.e., entry in the dataset) should hold metadata on a paper. Mandatory metadata are only `title` or `abstract`. If both title and abstract are available, the text is combined and used for training the model. If the column `title` is empty, the software will search for the next column `primary_title` and the same holds for `abstract` and `abstract_note`.

**Keywords, Authors** If `keywords` and/or `author` (or if the column is empty: `author names` or `first_authors`) are available it can be used for searching prior knowledge. Note the information is not shown during the screening phase and is also not used for training the model, but the information is available via the API.

**DOI and URL** If a Digital Object Identifier ( `DOI`) is available it will be displayed during the screening phase as a clickable hyperlink to the full text document. Similary, if a URL is provided, this is also displayed as a clickable link. Note by using ASReview you do *not* automatically have access to full-text and if you do not have access you might want to read this blog post.

**Included** A binary variable indicating the existing labeling decisions with `0 =` irrelevant/excluded, or `1 =` relevant/included. If no label is present, we assume the record is not seen by the reviewer. Different column names are allowed, see the table. The behavior of the labels is different for each mode, see *Fully, partially, and unlabeled data*.

## 9.2 RIS file format

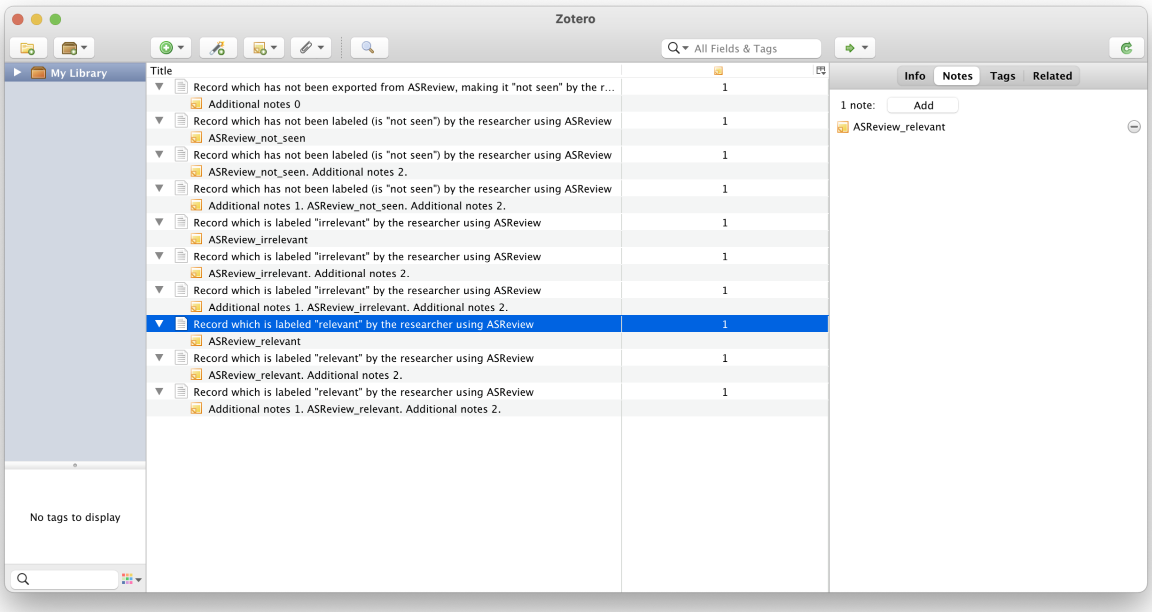
RIS file formats (with extensions `.ris` or `.txt`) are used by digital libraries, like IEEE Xplore, Scopus and ScienceDirect. Citation managers Mendeley, RefWorks, Zotero, and EndNote support the RIS file format as well. See (wikipedia) for detailed information about the format.

For parsing RIS file format, ASReview LAB uses a Python RIS files parser and reader (rispy). Successful import/export depends on a proper data set structure. The complete list of accepted fields and default mapping can be found on the rispy GitHub page.

The labels `ASReview_relevant`, `ASReview_irrelevant`, and `ASReview_not_seen` are stored with the N1 (Notes) tag, and can be re-imported into ASReview LAB. The behavior of the labels is different for each mode, see *Fully, partially, and unlabeled data*.

**Tip:** The labels `ASReview_relevant`, `ASReview_irrelevant`, and `ASReview_not_seen` are stored with the N1 (Notes) tag. In citation managers Zotero and Endnote the labels can be used for making selections; see the screenshots or watch the instruction video.

**Note:** When re-importing a partly labeled dataset in the RIS file format, the labels stored in the N1 field are used as prior knowledge. When a completely labeled dataset is re-imported it can be used in the Exploration and Simulation mode.

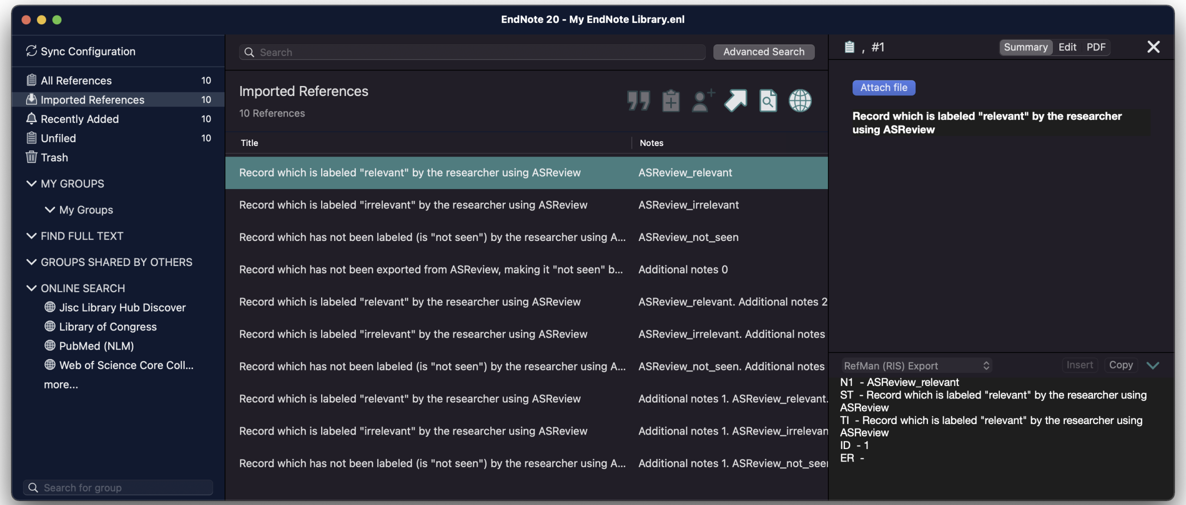


Title

Record which has not been exported from ASReview, making it "not seen" by the r... 1
Additional notes 0
Record which has not been labeled (is "not seen") by the researcher using ASReview 1
ASReview_not_seen
Record which has not been labeled (is "not seen") by the researcher using ASReview 1
ASReview_not_seen. Additional notes 2.
Record which has not been labeled (is "not seen") by the researcher using ASReview 1
Additional notes 1. ASReview_not_seen. Additional notes 2.
Record which is labeled "irrelevant" by the researcher using ASReview 1
ASReview_irrelevant
Record which is labeled "irrelevant" by the researcher using ASReview 1
ASReview_irrelevant. Additional notes 2.
Record which is labeled "irrelevant" by the researcher using ASReview 1
Additional notes 1. ASReview_irrelevant. Additional notes 2.
Record which is labeled "relevant" by the researcher using ASReview 1
ASReview_relevant
Record which is labeled "relevant" by the researcher using ASReview 1
ASReview_relevant. Additional notes 2.
Record which is labeled "relevant" by the researcher using ASReview 1
Additional notes 1. ASReview_relevant. Additional notes 2.

ASReview_relevant

Fig. 1: Example record with a labeling decision imported to Zotero



| Title | Notes |
|---|---|
| Record which is labeled "relevant" by the researcher using ASReview | ASReview_relevant |
| Record which is labeled "irrelevant" by the researcher using ASReview | ASReview_irrelevant |
| Record which has not been labeled (is "not seen") by the researcher using A... | ASReview_not_seen |
| Record which has not been exported from ASReview, making it "not seen" b... | Additional notes 0 |
| Record which is labeled "relevant" by the researcher using ASReview | ASReview_relevant. Additional notes 2 |
| Record which is labeled "irrelevant" by the researcher using ASReview | ASReview_irrelevant. Additional notes |
| Record which has not been labeled (is "not seen") by the researcher using A... | ASReview_not_seen. Additional notes |
| Record which is labeled "relevant" by the researcher using ASReview | Additional notes 1. ASReview_relevant. |
| Record which is labeled "irrelevant" by the researcher using ASReview | Additional notes 1. ASReview_irrelevan |
| Record which has not been labeled (is "not seen") by the researcher using A... | Additional notes 1. ASReview_not_seen |

**Record which is labeled "relevant" by the researcher using ASReview**

```
N1  - ASReview_relevant
ST  - Record which is labeled "relevant" by the researcher using
ASReview
TI  - Record which is labeled "relevant" by the researcher using
ASReview
ID  - 1
ER  -
```

Fig. 2: Example record with a labeling decision imported to Endnote

CHAPTER

TEN

# FULLY, PARTIALLY, AND UNLABELED DATA

Fully and partially labeled datasets serve a special role in the ASReview context. These datasets have review decisions for a subset of the records or for all records in the dataset.

## 10.1 Label format

For tabular datasets (*e.g., CSV, XLSX*), the dataset should contain a column called "included" or "label" (See *Data format* for all naming conventions), which is filled with `1`'s or `0`'s for the records that are already screened. The value is left empty for the records that you haven't screened yet, or which are added to the dataset in case of updating a review. For the RIS file format, the labels `ASReview_relevant`, `ASReview_irrelevant`, and `ASReview_not_seen`) can be stored with the N1(Notes) tag.

Exported files containing labeling decisions can be re-imported into ASReview LAB whereafter all labels are recognized and its behavior is different for each mode:

- In **Oracle mode** existing labels are used for prior knowledge.
- In **Validation mode** records are presented along with an indication of their previous labeling status: relevant, irrelevant, or not seen. This status is displayed via a color-coded bar above each record.
- In **Simulation** the column containing the labels is used to simulate a systematic review.

## 10.2 Unlabeled data

Unlabeled datasets do not contain any labels and can be used in the **Oracle mode** to start a review from scratch. Prior knowledge has to be selected in the *Prior Knowledge* step of the project set-up.

## 10.3 Partially labeled data

Partially labeled datasets are datasets with a labeling decision for a subset of the records in the dataset and no decision for another subset.

In **Oracle mode**, if labels are available for a part of the dataset, the labels will be automatically detected and used for *Prior Knowledge*. The first iteration of the model will then be based on these decisions and used to predict relevance scores for the unlabeled part of the data. It is usefull when a large number of records is needed for training, or when updating a systematic review, or to continue the screening process with model switching.

In **Validation mode**, the labels available are presented in the review screen along with an indication of their previous labeling status: relevant, irrelevant, or not seen. This status is displayed via a color-coded bar above each record, and you have the opportunity to refine the dataset by correcting any potential misclassifications, useful for the quality evaluation(see, for example, the SAFE procedure).

**Note:** Merging labeled with unlabeled data should be done outside ASReview LAB, for example, with the compose function of ASReview Datatools, or via *Citation Managers*.

## 10.4 Fully labeled data

Fully labeled datasets are datasets with a labeling decision for all records in the dataset.

In **Simulation mode**, the labels are used for mimicking the review proces for a *Simulation study*. Only records containing labels are used for the simulation, unlabeled records are ignored.

In **Validation mode**, the labels available in a fully labeled dataset are presented in the review screen along with an indication of their previous labeling status: relevant or irrelevant. It is usefull to validate labels as a human when the labels are predicted by a large language model (LLM), like by ChatGPT. Also, one can use this mode for teaching purporses.

### 10.4.1 Benchmark datasets

The ASReview research project collects fully labeled datasets published open access. The labeled datasets are PRISMA-based systematic reviews or meta-analyses on various research topics. They can be useful for teaching purposes or for testing the performance of (new) active learning models. The datasets and their metadata are available via the SYNERGY Dataset repository. In ASReview LAB, these datasets are found under "Benchmark Datasets"; only available for the Validation and Simulation modi.

The Benchmark Datasets are directly available in the software. During the *Add dataset* step of the project setup, there is a panel with all the datasets. The datasets can be selected and used directly. Benchmark datasets are also available via the *Simulation via command line*. Use the prefix `synergy:` followed by the identifier of the dataset (see Synergy Dataset repository). For example, to use the Van de Schoot et al. (2018) dataset, use `synergy:van_de_schoot_2018`.

CHAPTER

ELEVEN

# CREATE A PROJECT

To start reviewing a dataset with ASReview LAB, you first need to create a project. The project will contain your dataset, settings, labeling decisions, and machine learning models. You can choose from three different project types: Oracle, Validation, and Simulation. The project setup consists of 4 steps: Project information, Data, Model, and Warm up. The sections below explain each of the steps of the setup.

To create a project:

1. *Start ASReview LAB*.
2. Go to the *Projects dashboard* if you are not already there (http://localhost:5000/projects)
3. Click on the *Create* button on the bottom left

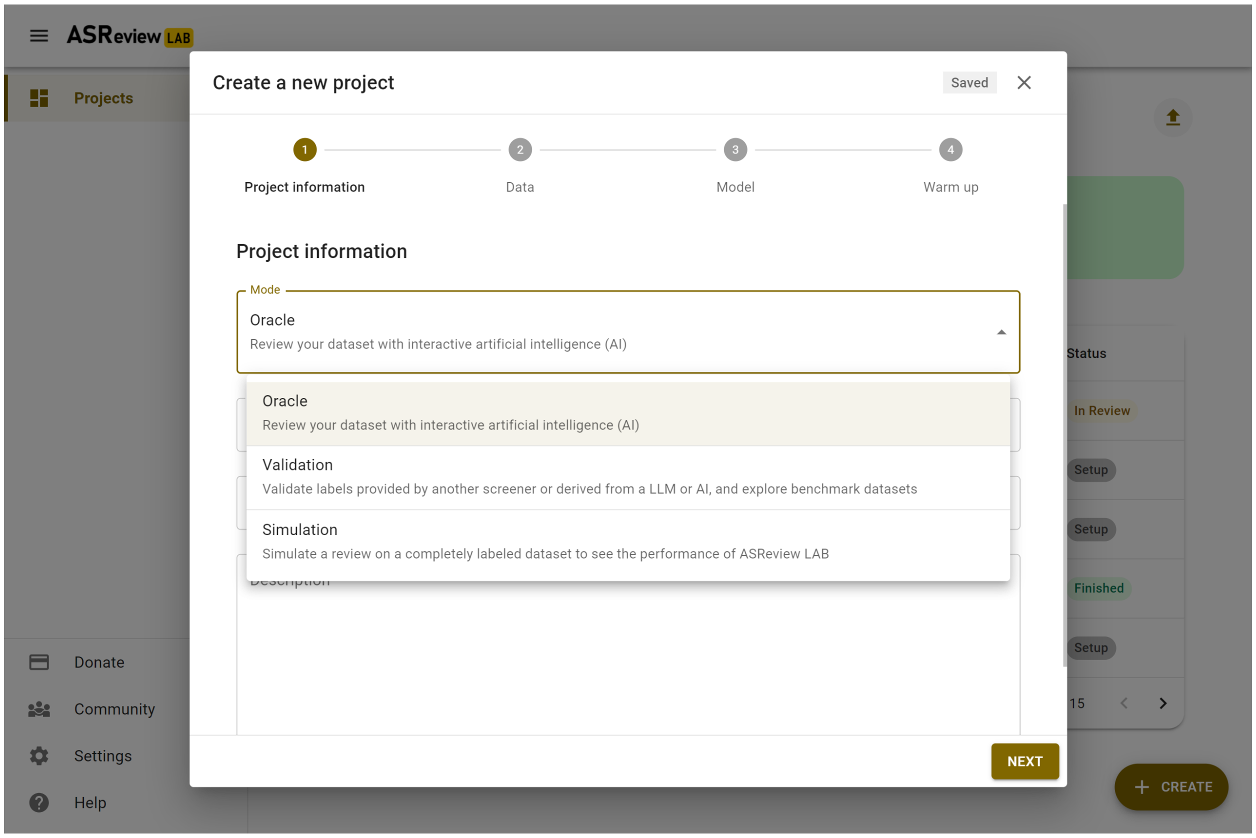
## 11.1 Project information

In Step 1, you provide all relevant information about your project as well as the type of project you want (the mode). The sections below provide more information on the input fields. After you complete this step, click *next*.

### 11.1.1 Project modes

In this step, you have to select a mode. The default is **Oracle**. For a description of all modi, see *Fully, partially, and unlabeled data*. In short, if you want to:

- screen a dataset from scratch -> Oracle mode with unlabeled data;
- continue screening, for example using a different model -> Oracle mode with partly labeled data;
- validate labels provided by a another screener or predicted by a Large Language Model (e.g., ChatGPT) -> Validation mode with partly or fully labeled data;
- learn how the software with active learning works -> Validation mode with fully labeled data;
- mimic the screening process in a simulation study -> Simulation mode with fully labeled data.


ASReview LAB
Projects
Create a new project
Saved
1 Project information
2 Data
3 Model
4 Warm up
Project information
Mode
Oracle
Review your dataset with interactive artificial intelligence (AI)
Oracle
Review your dataset with interactive artificial intelligence (AI)
Validation
Validate labels provided by another screener or derived from a LLM or AI, and explore benchmark datasets
Simulation
Simulate a review on a completely labeled dataset to see the performance of ASReview LAB
Description
NEXT
Status
In Review
Setup
Setup
Finished
Setup
Donate
Community
Settings
Help
CREATE

### 11.1.2 Project details

Provide project details like name of the project (required), author(s) (for example, the name of the screener), and a description. You can edit these values later in the *Details* page.

## 11.2 Data and Prior Knowledge

In Step 2, you import a dataset and select prior knowledge.

### 11.2.1 Add dataset

Click on *Add* to select a dataset. The data needs to adhere to a *specific format*. Keep in mind that in Oracle mode, your dataset is unlabeled or *Partially labeled data*; in Validation mode *Partially labeled data* or fully labeled; and for Simulation mode, you need *Fully labeled data*.

**Tip:** You will benefit most from what active learning has to offer with *High-quality data*.

Depending on the *Project mode*, you are offered different options for adding a dataset:

#### From File

Drag and drop your file or select your file. Click on *Save* on the top right.

**Note:** After adding your dataset, ASReview LAB shows the approximate number of duplicates. This number is based on duplicate titles and abstracts and if available, on the Digital Object Identifier (DOI). Removing duplicates can be done via ASReview Datatools, which also allows using a persistent identifier (PID) other than DOI for identifying and removing duplicates.

#### From URL or DOI

Insert a URL to a dataset. For example, use a URL from this dataset repository. It is also possible to provide a DOI to a data repository (supported for many data repositories via Datahugger). In a DOI points to multiple files, select the file you want to use (e.g. 10.17605/OSF.IO/WDZH5).

Click on *Add* to add the dataset.

#### From Extension

Select a file available via an extension (Oracle and Validation only). Click on *Save* on the top right.

#### Benchmark Datasets

Select one of the *Benchmark datasets* (Simulation and Validation only). Click on *Save* on the top right.

### 11.2.2 Prior Knowledge

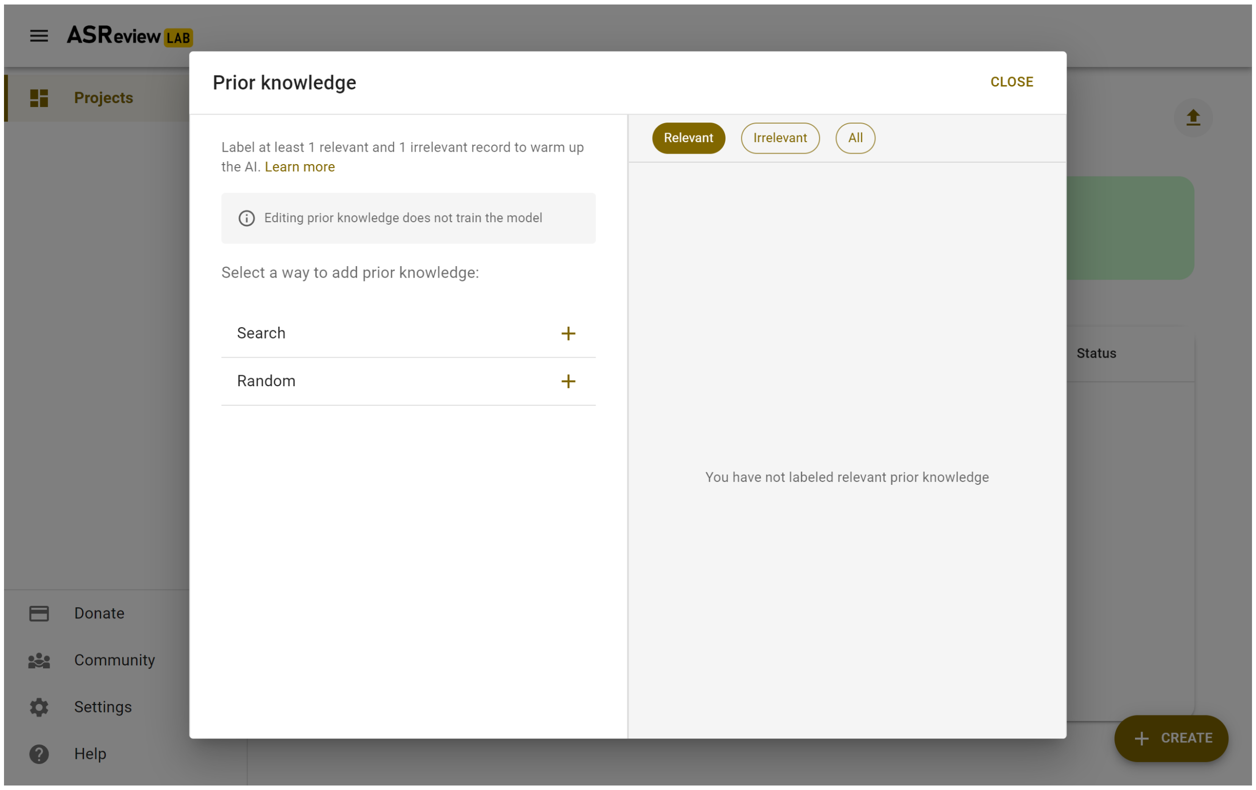
The first iteration of the active learning cycle requires training data, referred to as prior knowledge. This knowledge is used by the classifier to create an initial ranking of the unseen records. In this step, you need to provide a minimum training data set of size two, with **at least** one relevant and one irrelevant labeled record.

**Note:** If you use *Partially labeled data* in the Oracle mode, you can skip this step, because the labels available in the dataset are used for training the first iteration of the model.

To facilitate prior selection, it is possible to search within your dataset, or . This is especially useful for finding records that are relevant based on previous studies or expert consensus.

You can also let ASReview LAB present you with random records. This can be useful for finding irrelevant records.

The interface works as follows; on the left, you will see methods to find records to use as prior knowledge, on the right, you will see your selected prior knowledge. If you have **at least** one relevant and one irrelevant record, you can click *Close* and go to the next step.



Label at least 1 relevant and 1 irrelevant record to warm up the AI. Learn more

Editing prior knowledge does not train the model

Select a way to add prior knowledge:

Search

Random

You have not labeled relevant prior knowledge

## Search

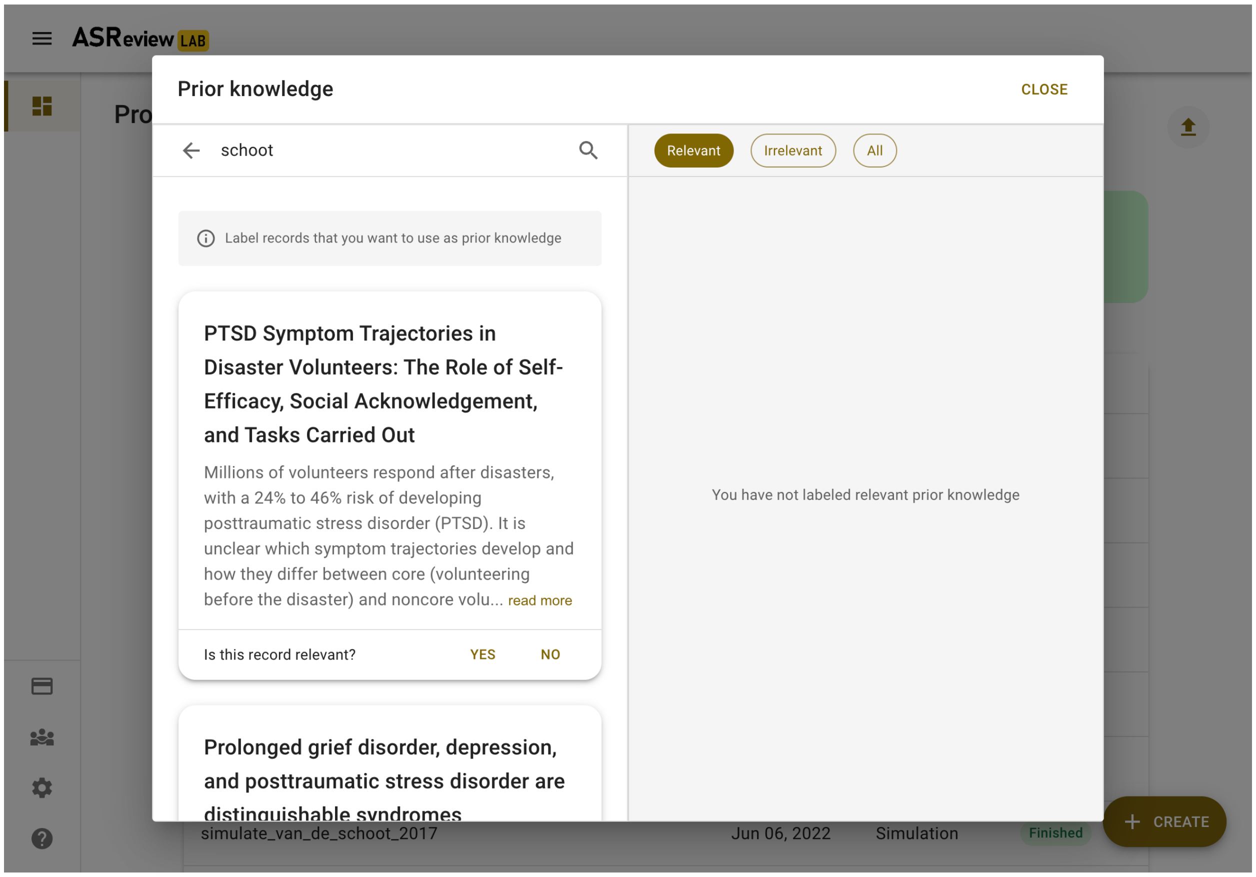
Let's start with finding a prior relevant document. The most efficient way to do this is by searching for a specific document that you already know is relevant. Click on Search and search your dataset by authors, keywords or title, or a combination thereof. Make sure to be precise with the search terms, as only the first 10 results are shown to you. After entering your search terms, press enter to start searching.

Click the document you had in mind and answer, "Is this record relevant?". Note, don't label all items here. Only the one you are looking for and want to use as training data.

The prior knowledge will now show up on the right. There are no restrictions on the number of records and the software already works with 2 labels (1 relevant and 1 irrelevant).

The prior knowledge will now show up on the right. Use the buttons to see all prior knowledge or a subset. You can also change the label or remove the record from the training set. There are no restrictions on the number of records you provide, and the software already works with 2 labeled records (1 relevant and 1 irrelevant). After labeling five randomly selected records, ASReview LAB will ask you whether you want to stop searching prior knowledge. Click on *STOP* and click *Next*.

Inspect the records to be used for training the first iteration of the model, and if you are done, click *Close*.



Label records that you want to use as prior knowledge

PTSD Symptom Trajectories in Disaster Volunteers: The Role of Self-Efficacy, Social Acknowledgement, and Tasks Carried Out

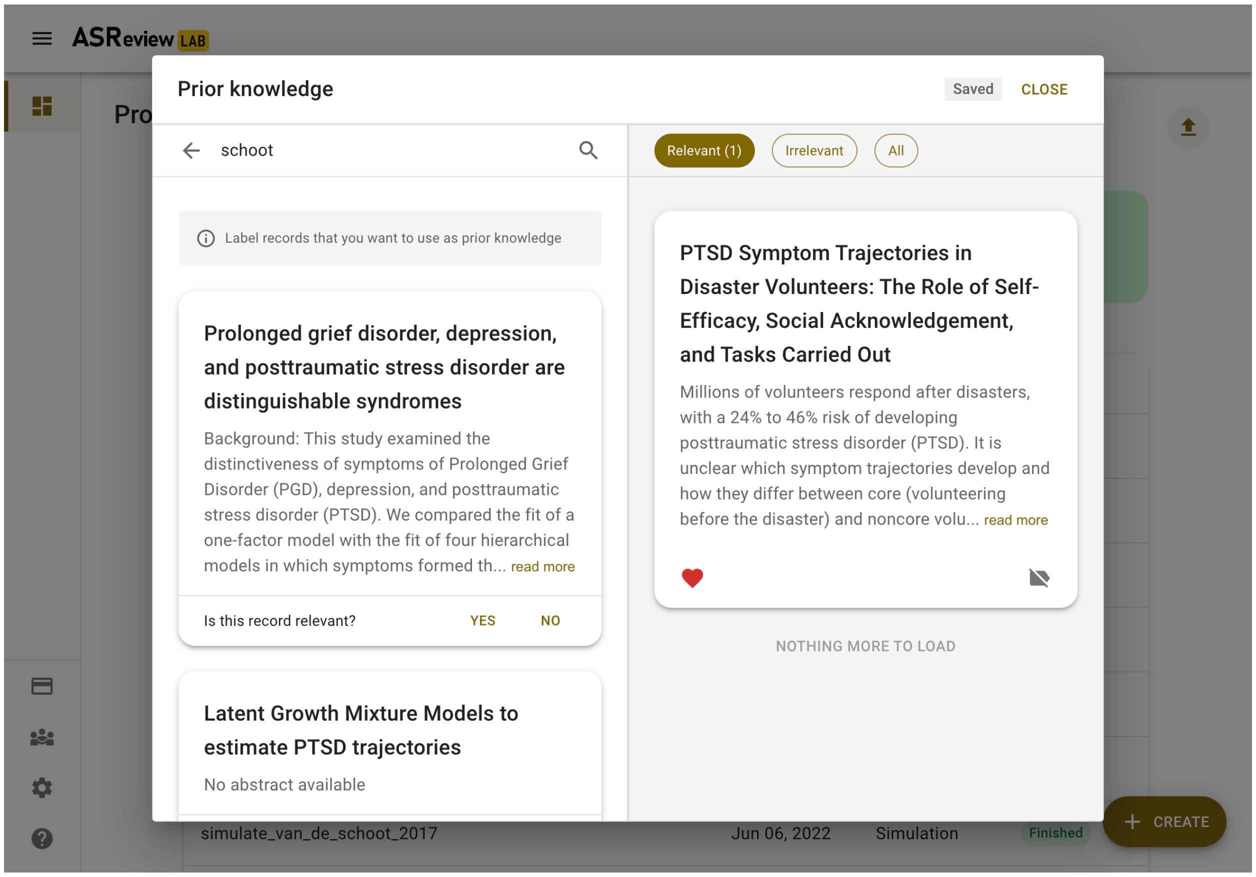
Millions of volunteers respond after disasters, with a 24% to 46% risk of developing posttraumatic stress disorder (PTSD). It is unclear which symptom trajectories develop and how they differ between core (volunteering before the disaster) and noncore volu... read more

Is this record relevant? YES NO

Prolonged grief disorder, depression, and posttraumatic stress disorder are distinguishable syndromes

You have not labeled relevant prior knowledge

simulate_van_de_schoot_2017 Jun 06, 2022 Simulation Finished



Label records that you want to use as prior knowledge

**Prolonged grief disorder, depression, and posttraumatic stress disorder are distinguishable syndromes**

Background: This study examined the distinctiveness of symptoms of Prolonged Grief Disorder (PGD), depression, and posttraumatic stress disorder (PTSD). We compared the fit of a one-factor model with the fit of four hierarchical models in which symptoms formed th... read more

Is this record relevant? YES NO

**Latent Growth Mixture Models to estimate PTSD trajectories**

No abstract available

**PTSD Symptom Trajectories in Disaster Volunteers: The Role of Self-Efficacy, Social Acknowledgement, and Tasks Carried Out**

Millions of volunteers respond after disasters, with a 24% to 46% risk of developing posttraumatic stress disorder (PTSD). It is unclear which symptom trajectories develop and how they differ between core (volunteering before the disaster) and noncore volu... read more

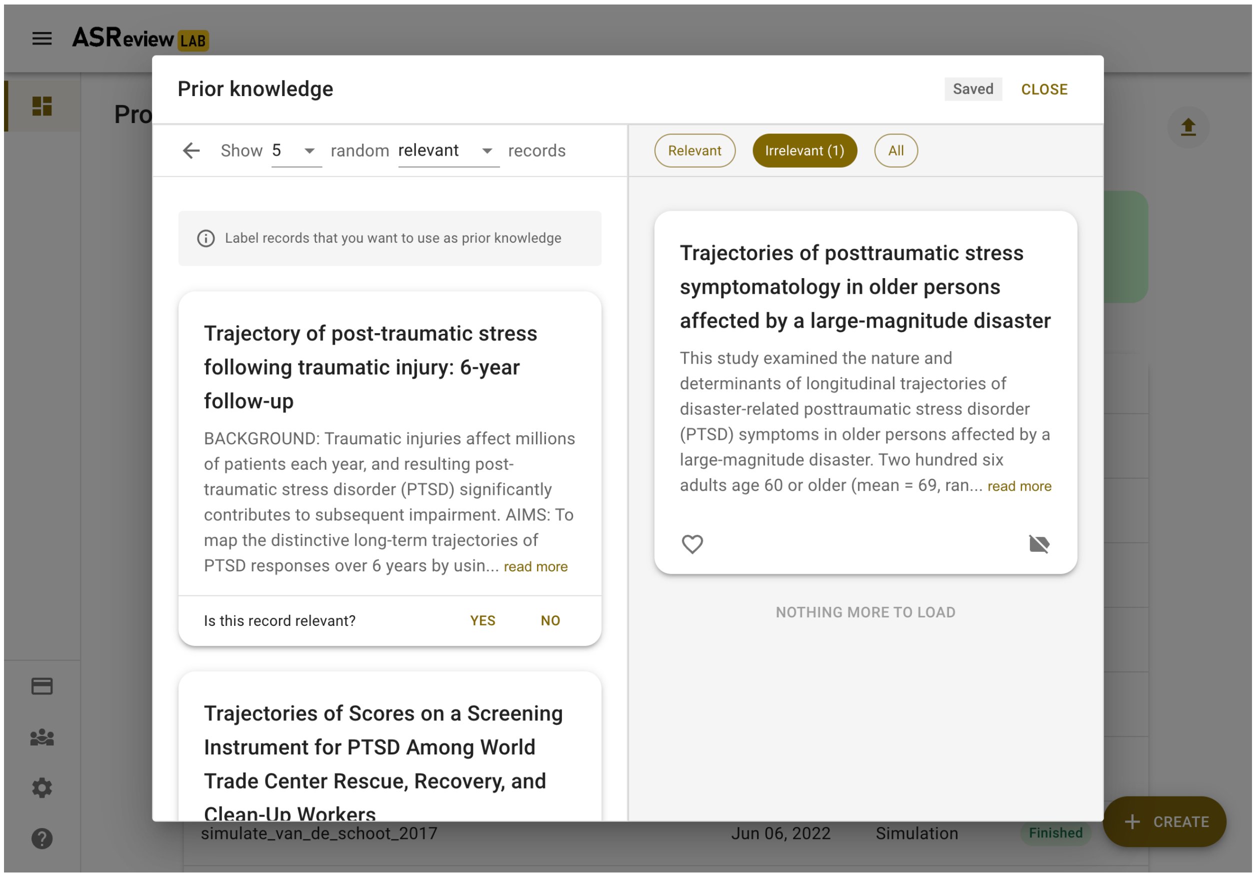
NOTHING MORE TO LOAD

simulate_van_de_schoot_2017 Jun 06, 2022 Simulation Finished

### Random

**Warning:** Do not use the random option to search for the sparse relevant records!

You also need to provide at least one prior irrelevant document. One way to find an irrelevant document is by labeling a set of random records from the dataset. Given that the majority of records in the dataset are irrelevant (extremely imbalanced data problem), the records presented here are likely to be irrelevant for your study. Click on *random* to show a few random records. Indicate for each record you want to use as training data whether it is irrelevant (or relevant).



Trajectory of post-traumatic stress following traumatic injury: 6-year follow-up

BACKGROUND: Traumatic injuries affect millions of patients each year, and resulting post-traumatic stress disorder (PTSD) significantly contributes to subsequent impairment. AIMS: To map the distinctive long-term trajectories of PTSD responses over 6 years by usin...

Trajectories of Scores on a Screening Instrument for PTSD Among World Trade Center Rescue, Recovery, and Clean-Up Workers

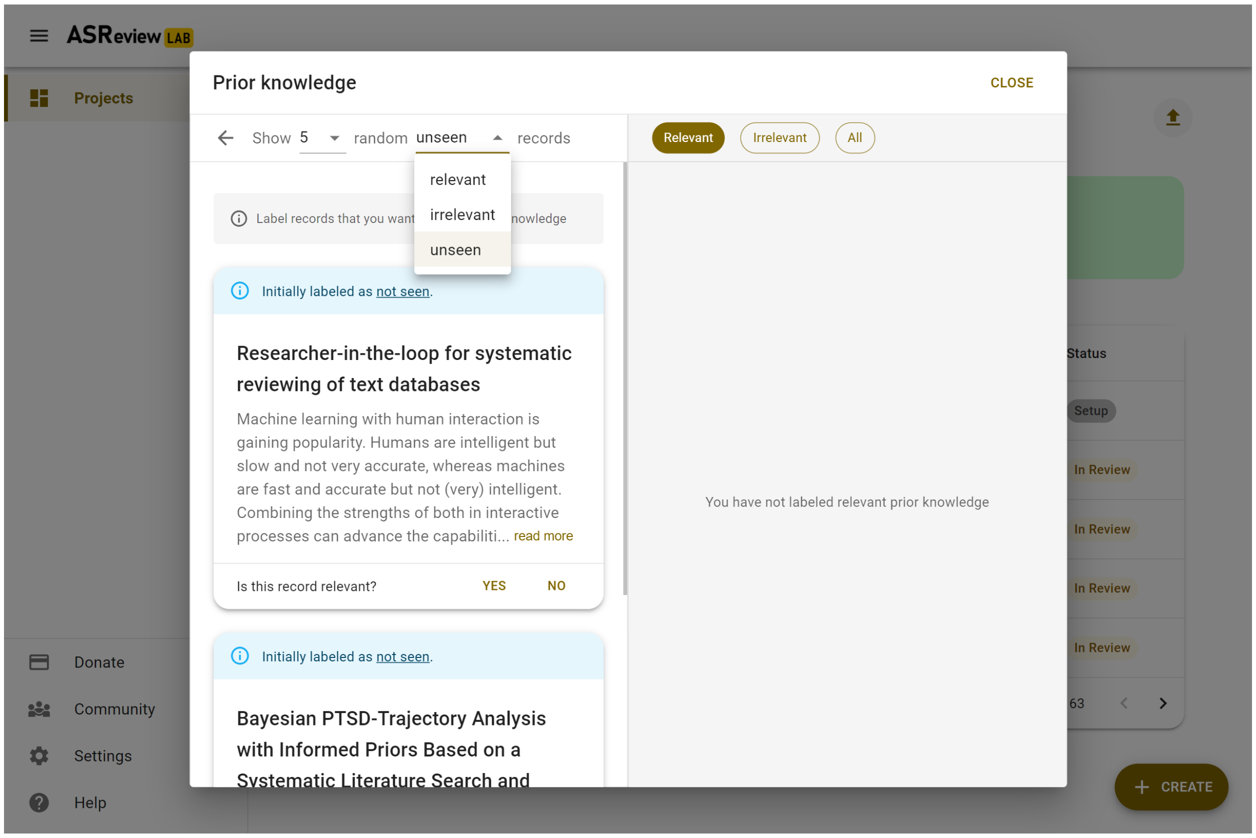
Trajectories of posttraumatic stress symptomatology in older persons affected by a large-magnitude disaster

This study examined the nature and determinants of longitudinal trajectories of disaster-related posttraumatic stress disorder (PTSD) symptoms in older persons affected by a large-magnitude disaster. Two hundred six adults age 60 or older (mean = 69, ran...

In the Validation mode when selecting random records, one can choose random records from the subset of initially labeled relevant, irrelevant or not seen records. The initial labels are displayed via a color-coded bar above each record.

## 11.3 Model

In the next step of the setup, you can select the active learning model. The default settings (Naïve Bayes, TF-IDF, Max) have fast and excellent performance. Most users can skip this step and click *Next*. More information about the active learning process can be found in the blog post Active learning explained,


ASReview LAB
Projects
Prior knowledge
CLOSE
Show 5 random unseen records
relevant
irrelevant
unseen
Relevant
Irrelevant
All
Label records that you want ... nowledge
Initially labeled as not seen.
Researcher-in-the-loop for systematic reviewing of text databases
Machine learning with human interaction is gaining popularity. Humans are intelligent but slow and not very accurate, whereas machines are fast and accurate but not (very) intelligent. Combining the strengths of both in interactive processes can advance the capabiliti... read more
Is this record relevant?
YES
NO
Initially labeled as not seen.
Bayesian PTSD-Trajectory Analysis with Informed Priors Based on a Systematic Literature Search and
You have not labeled relevant prior knowledge
Status
Setup
In Review
In Review
In Review
In Review
63
CREATE
Donate
Community
Settings
Help

### 11.3.1 Select model

It is possible to change the settings of the Active learning model. There are four settings that can be changed in the software:

#### Feature extraction

The feature extraction technique determines the method how text is translated into a vector that can be used by the classifier. The default is TF-IDF (Term Frequency-Inverse Document Frequency) from SKLearn. It works well in combination with Naive Bayes and other fast training models.

Another recommended option is Doc2Vec provided by the gensim package. Before starting ASReview LAB, first, install *gensim*:

```
pip install asreview[gensim]
```

**Note:** It takes relatively long to create a feature matrix with Doc2Vec, but this only has to be done once. The upside of this method is that it takes context into account. Also, a benefit is the dimension-reduction that generally takes place, which makes the modeling quicker.

Several other feature extractors are available in the software (sentence Bert, embedding IDF/LSTM) and more classifiers can be selected via the *API*, or added via *Extensions*.

### Classifier

The classifier is the machine learning model used to compute the relevance scores. The default is Naive Bayes. Though relatively simplistic, it seems to work quite well on a wide range of datasets. Several other classifiers are available in the software (logistic regression, random forest, SVM, LSTM, neural net) and more classifiers can be selected via the *API* or added via *Extensions*.

The neural nets require tensorflow, use

```
pip install asreview[tensorflow]
```

### Balancing Strategy

To decrease the class imbalance in the training data, the default is to rebalance the training set by a technique called dynamic resampling (DR) (Ferdinands et al., 2020). DR undersamples the number of irrelevant records in the training data, whereas the number of relevant records are oversampled such that the size of the training data remains the same. The ratio between relevant and irrelevant records in the rebalanced training data is not fixed, but dynamically updated and depends on the number of records in the available training data, the total number of records in the dataset, and the ratio between relevant and irrelevant records in the available training data. No balancing or undersampling are the other options. Other strategies can be selected via the *API* or added via *Extensions*.

### Query Strategy

The query strategy determines which document is shown after the model has computed the relevance scores. The options are: maximum (certainty-based), uncertainty, random, and clustering. When certainty-based is selected, the documents are shown in the order of relevance score. The document most likely to be relevant is shown first. When mixed is selected, the next document will be selected certainty-based 95% of the time, and uncertainty based or randomly chosen otherwise. When random is selected, documents are shown in a random order (ignoring the model output completely). Other strategies can be selected via the *API* or added via *Extensions*.

**Warning:** Selecting *random* means your review will not be accelerated by using ASReview.

### Model switching

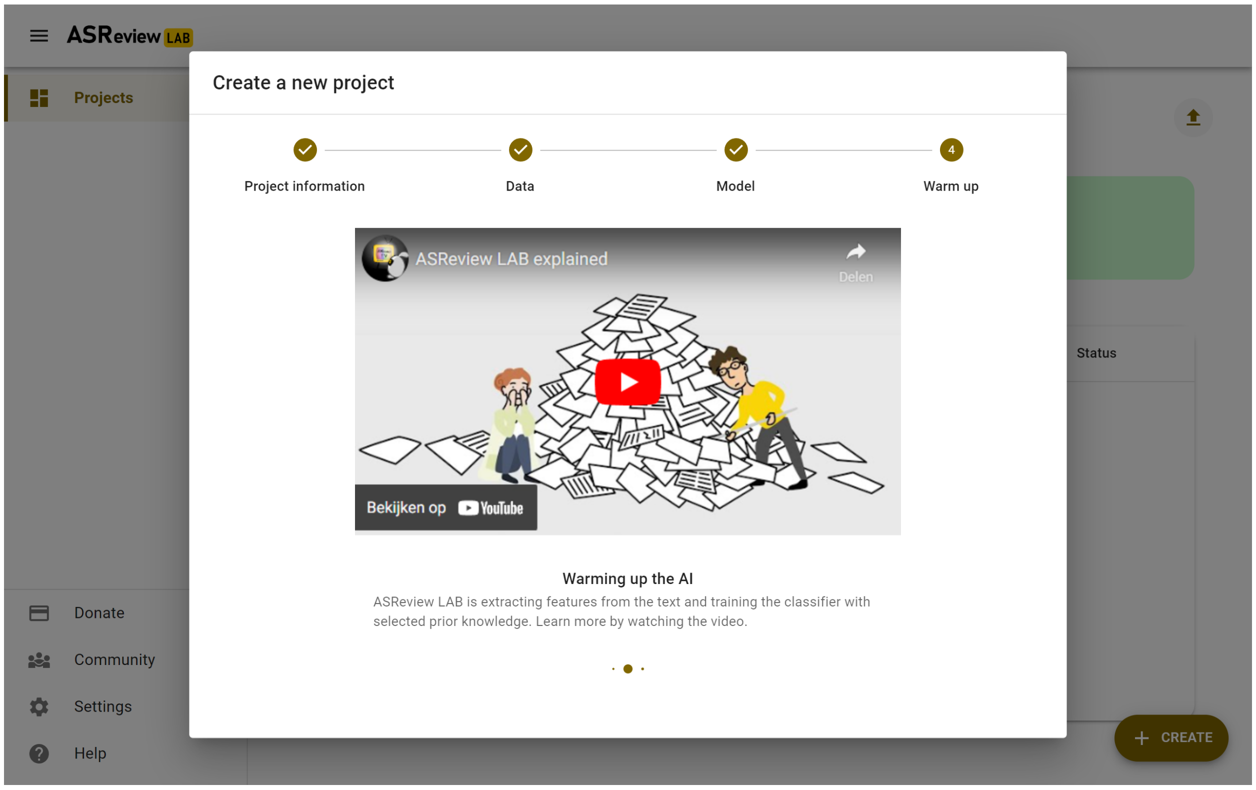
During the screening phase, it is not possible to change the model. However, it is possible to select a first model, screen part of the data, and export the dataset with the labeling decisions of the first model. This partly-labeled dataset can be imported into a new project and the labels based on the first model will be recognized as prior knowledge. Then, a second model can be trained on the partly-labeled data, and the new predictions will be based on the second model.

**Tip:** It is suggested to screen with a simple active learning model (e.g., the defaults) first until you reach your stopping criteria, then switch to a different model (e.g., doc2vec plus a neural net) and screen again until you reach your stopping criteria.

## 11.4 Warm up

In the last step of the setup, step 4, ASReview LAB runs the feature extractor and trains a model, and ranks the records in your dataset. Depending on the model and the size of your dataset, this can take a couple of minutes (or even longer; you can enjoy the animation video). After the project is successfully initialized, you can start reviewing.

**Note:** In Simulation mode, this step starts the simulation. As simulations usually take longer to complete, the simulation will run in the background. After a couple of seconds, you will see a message and a button "Got it". You will navigate to the *Analytics* page, where you can follow the progress (see *Refresh* button on the top right)



Warming up the AI

ASReview LAB is extracting features from the text and training the classifier with selected prior knowledge. Learn more by watching the video.

CHAPTER

TWELVE

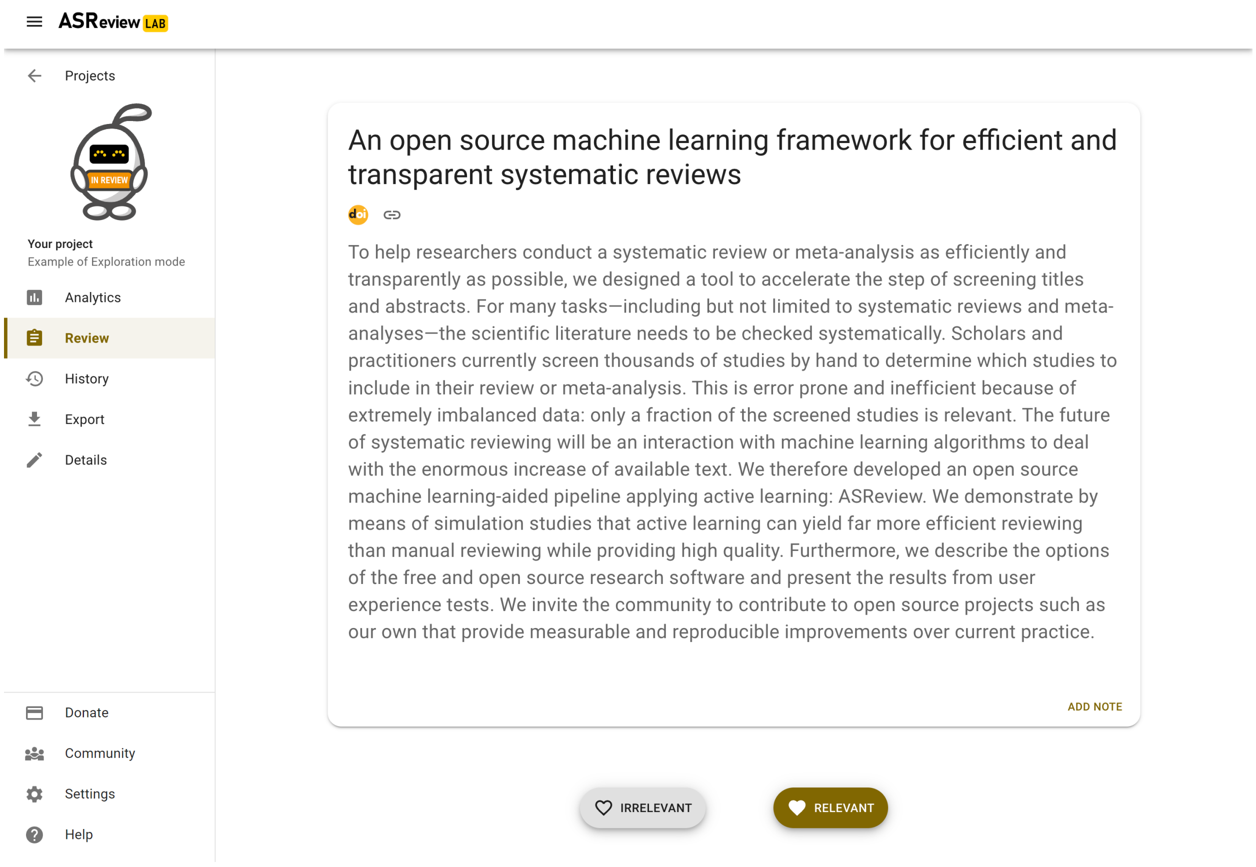
# SCREENING

**Note:** Only for Oracle and Validation. Read more about the options for the Simulation mode in the *Overview*.

## 12.1 Introduction

As soon as your project is initiated, you can start reviewing. Click on *Review* in the left menu if your project is not on the review page yet. ASReview LAB presents you a title and abstract to screen and label.

You are asked to make a decision: relevant or irrelevant?



An open source machine learning framework for efficient and transparent systematic reviews

To help researchers conduct a systematic review or meta-analysis as efficiently and transparently as possible, we designed a tool to accelerate the step of screening titles and abstracts. For many tasks—including but not limited to systematic reviews and meta-analyses—the scientific literature needs to be checked systematically. Scholars and practitioners currently screen thousands of studies by hand to determine which studies to include in their review or meta-analysis. This is error prone and inefficient because of extremely imbalanced data: only a fraction of the screened studies is relevant. The future of systematic reviewing will be an interaction with machine learning algorithms to deal with the enormous increase of available text. We therefore developed an open source machine learning-aided pipeline applying active learning: ASReview. We demonstrate by means of simulation studies that active learning can yield far more efficient reviewing than manual reviewing while providing high quality. Furthermore, we describe the options of the free and open source research software and present the results from user experience tests. We invite the community to contribute to open source projects such as our own that provide measurable and reproducible improvements over current practice.

### 12.1.1 Screening in Oracle mode

In the Oracle mode, unlabeled records are presented to you. Depending on the selected strategy it is the most likely relevant record (default setting) or based on another:ref:project_create:Query Strategy.

Click on the decision of your choice, and a new record is presented to you. While you review the next record, a new model is being trained. ASReview LAB continuously improves its understanding of your decisions, constantly updating the underlying ordering of the records.

Each labeling decision of the user starts the training of a new model given no model is being trained at that time. When this new model is trained, the unseen records' rank order is updated. Training and labeling occur asynchronously. With fast models, a new ranking will probably be available before the user finishes reading the text. With slower models, training continues until a new model is trained, and the user can continue screening the next record in line (2nd, 3rd, etc.). Therefore, the record shown to the user can be the one with the highest relevance score of the second last model or the highest-ranked as a result of the latest model until a new model is trained. For a detailed description of the the data model, see the paper Reproducibility and Data Storage Checklist.

As you keep reviewing documents and providing labels, you will probably see fewer relevant records. When to stop screening is left to you. See *Progress and results* for more information on progress monitoring and information on when to stop.

---

**Tip:** If you are in doubt about your decision, take your time as you are the oracle. Based on your input, a new model will be trained, and you do not want to confuse the prediction mode. For the model, it may be better to consult others, and read the full text (in case of reviewing abstracts of scientific papers)

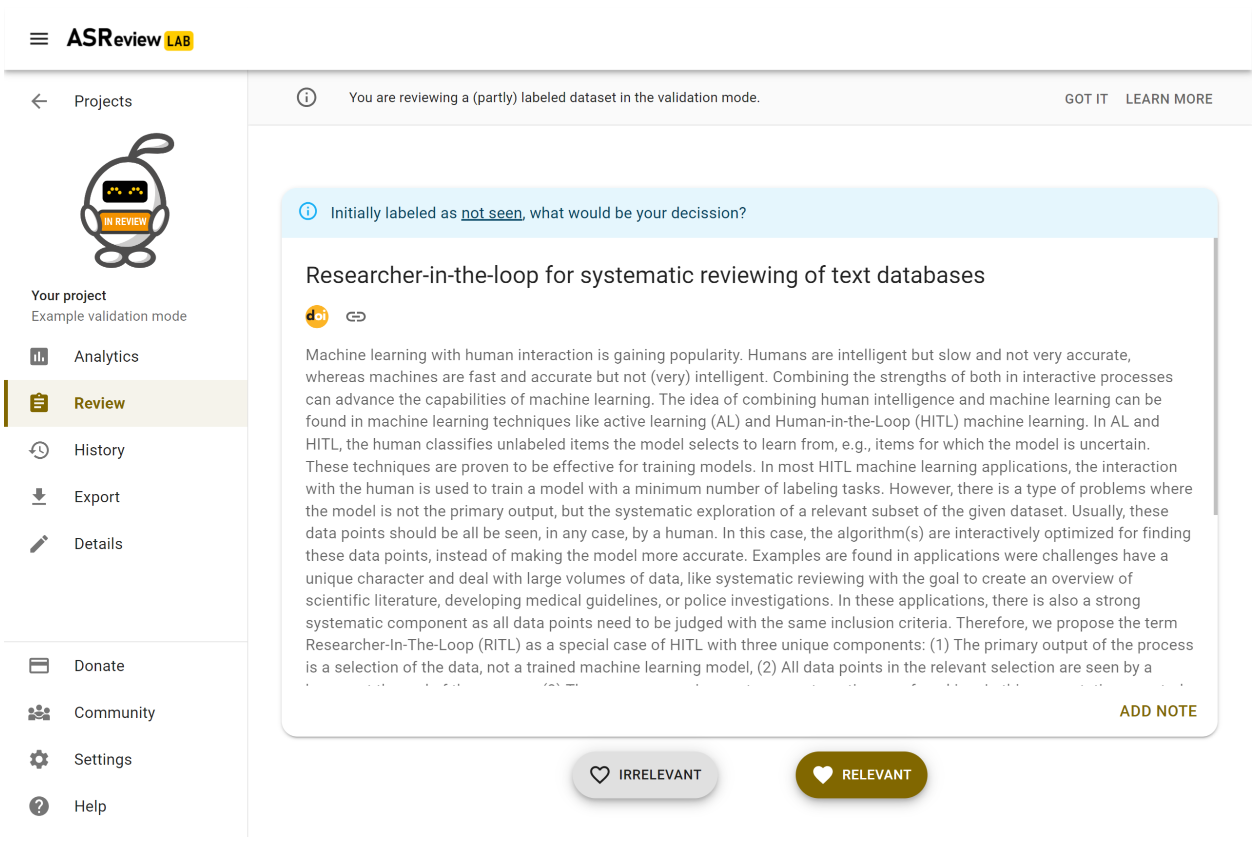
---

### 12.1.2 Screening in Validation mode

The Validation mode (formerly known as Exploration mode) is tailored for scenarios where it's necessary to validate existing labels or engage in a review process without being an oracle. This mode is especially beneficial for validating labels made by a first screener, reviewing labels predicted by Large Language Models(LLMs) such as ChatGPT, or for educational and training purposes.

In this mode, records are presented along with an indication of their previous labeling status: relevant, irrelevant, or not seen. This status is displayed via a color-coded bar above each record. If a record was labeled by another screener or an AI model, you have the opportunity to validate, or challenge these labels, helping to refine the dataset by correcting any potential misclassifications, useful for the quality evaluation of the SAFE procedure.

Additionally, the Validation mode is useful for educational use. Instructors and learners can utilize this mode to simulate the screening process without being the expert decision-maker. This setup is particularly advantageous in workshop settings, where participants can engage with the screening process using the labeled SYNERGY datasets. This hands-on experience offers valuable insights into the software's functionality and the systematic review process without the need to be a content expert. For comprehensive online teaching materials and tutorials on using ASReview LAB effectively, please visit the ASReview Academy.



You are reviewing a (partly) labeled dataset in the validation mode.

Initially labeled as not seen, what would be your decission?

Researcher-in-the-loop for systematic reviewing of text databases

Machine learning with human interaction is gaining popularity. Humans are intelligent but slow and not very accurate, whereas machines are fast and accurate but not (very) intelligent. Combining the strengths of both in interactive processes can advance the capabilities of machine learning. The idea of combining human intelligence and machine learning can be found in machine learning techniques like active learning (AL) and Human-in-the-Loop (HITL) machine learning. In AL and HITL, the human classifies unlabeled items the model selects to learn from, e.g., items for which the model is uncertain. These techniques are proven to be effective for training models. In most HITL machine learning applications, the interaction with the human is used to train a model with a minimum number of labeling tasks. However, there is a type of problems where the model is not the primary output, but the systematic exploration of a relevant subset of the given dataset. Usually, these data points should be all be seen, in any case, by a human. In this case, the algorithm(s) are interactively optimized for finding these data points, instead of making the model more accurate. Examples are found in applications were challenges have a unique character and deal with large volumes of data, like systematic reviewing with the goal to create an overview of scientific literature, developing medical guidelines, or police investigations. In these applications, there is also a strong systematic component as all data points need to be judged with the same inclusion criteria. Therefore, we propose the term Researcher-In-The-Loop (RITL) as a special case of HITL with three unique components: (1) The primary output of the process is a selection of the data, not a trained machine learning model, (2) All data points in the relevant selection are seen by a

## 12.2 Autosave

Your decisions (and notes) are saved automatically into your ASReview project file. There is no need to press any buttons to save your work anywhere in ASReview LAB (in fact, there is not even a *save* button).

## 12.3 Change decisions

In some cases, you might want to change your previous decision. The screening interface of ASReview LAB offers two options to change your decision.

### 12.3.1 Undo last decision

You can return to your previous decision during screening.

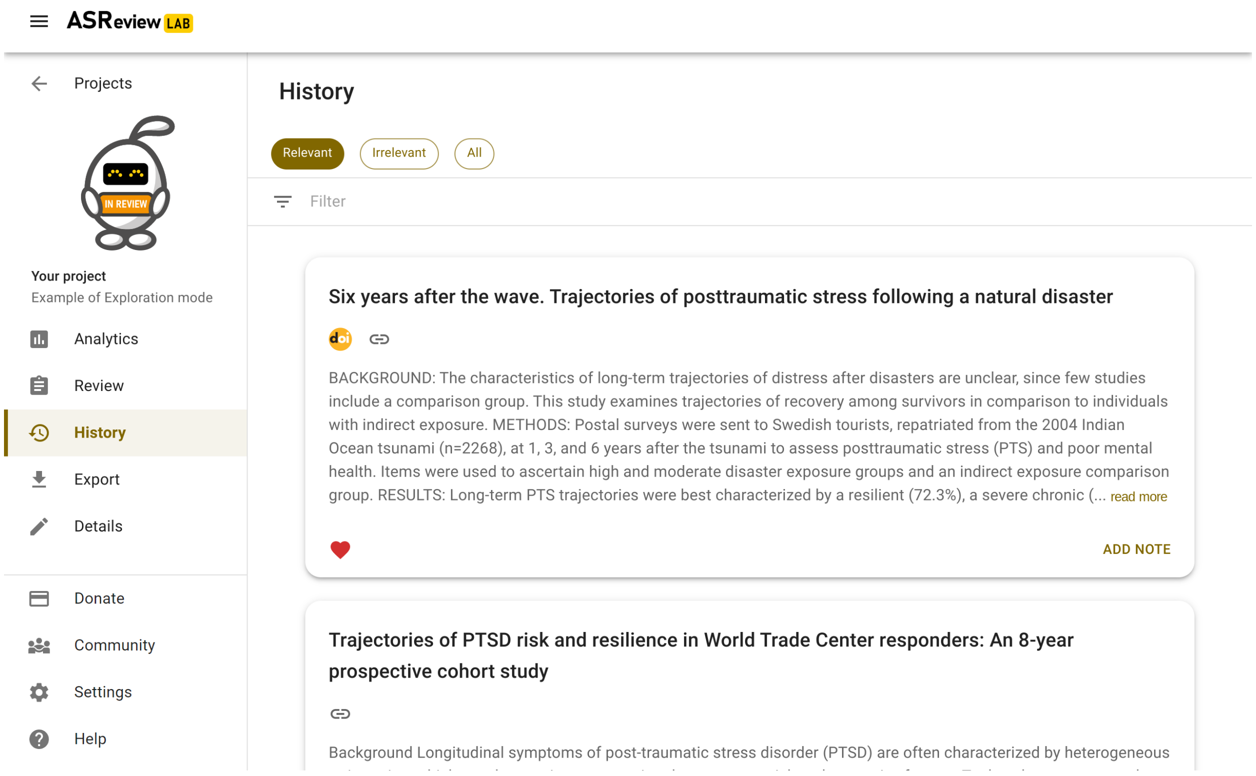
1. *Start ASReview LAB*.
2. Open or *Create a project*.
3. Label the record displayed in the screen as relevant or irrelevant.
4. Click on *Undo* (At the bottom right)
5. Click on *Keep (ir)relevant* or *Convert to (ir)relevant*.
6. Continue screening.

You can disable this option in the Settings menu.

### 12.3.2 Screening history

An overview of your decisions made during screening can be found on the **History** page. You can change decisions on this page.

1. *Start ASReview LAB*.
2. Open or *Create a project*.
3. Click on History in the menu on the left.



Six years after the wave. Trajectories of posttraumatic stress following a natural disaster

BACKGROUND: The characteristics of long-term trajectories of distress after disasters are unclear, since few studies include a comparison group. This study examines trajectories of recovery among survivors in comparison to individuals with indirect exposure. METHODS: Postal surveys were sent to Swedish tourists, repatriated from the 2004 Indian Ocean tsunami (n=2268), at 1, 3, and 6 years after the tsunami to assess posttraumatic stress (PTS) and poor mental health. Items were used to ascertain high and moderate disaster exposure groups and an indirect exposure comparison group. RESULTS: Long-term PTS trajectories were best characterized by a resilient (72.3%), a severe chronic (...

Trajectories of PTSD risk and resilience in World Trade Center responders: An 8-year prospective cohort study

Background Longitudinal symptoms of post-traumatic stress disorder (PTSD) are often characterized by heterogeneous

Changing decisions on the history page

4. To change a label of a record, click the heart icon. The next iteration of the model will take the new label into account.

## 12.4 Full Text

If a column with Digital Object Identifiers (DOI) or URLs is available in the metadata of your dataset, ASReview LAB will display the DOI and URL during screening. Most of the time, DOIs point to the full-text of a publication. See *datasets* for more information on including DOI and URL values to your datasets.

## 12.5 Keyboard shortcuts

ASReview LAB supports the use of keyboard shortcuts during screening. The table below lists the available keyboard shortcuts.

You can press a key (or a combination of keys) to label a record as relevant or irrelevant, or to return to the previous decision during screening. By default, keyboard shortcuts are disabled.

| Action | Shortcut |
|---|---|
| Label record as relevant | **r** or **Shift + r** |
| Label record as irrelevant | **i** or **Shift + i** |
| Return to previous decision | **u** or **Shift + u** |

**Note:** Keyboard shortcuts are only available when the **Undo** feature has been enabled in the Settings (bottom left).

## 12.6 Display

### 12.6.1 Dark mode

ASReview LAB offers the option to customize the screening appearance and functionality.

1. *Start ASReview LAB*.
2. Click on *Settings* (bottom left).
3. Go to *Display* and toggle the dark mode

**Note:** Your preference is saved in the browser.

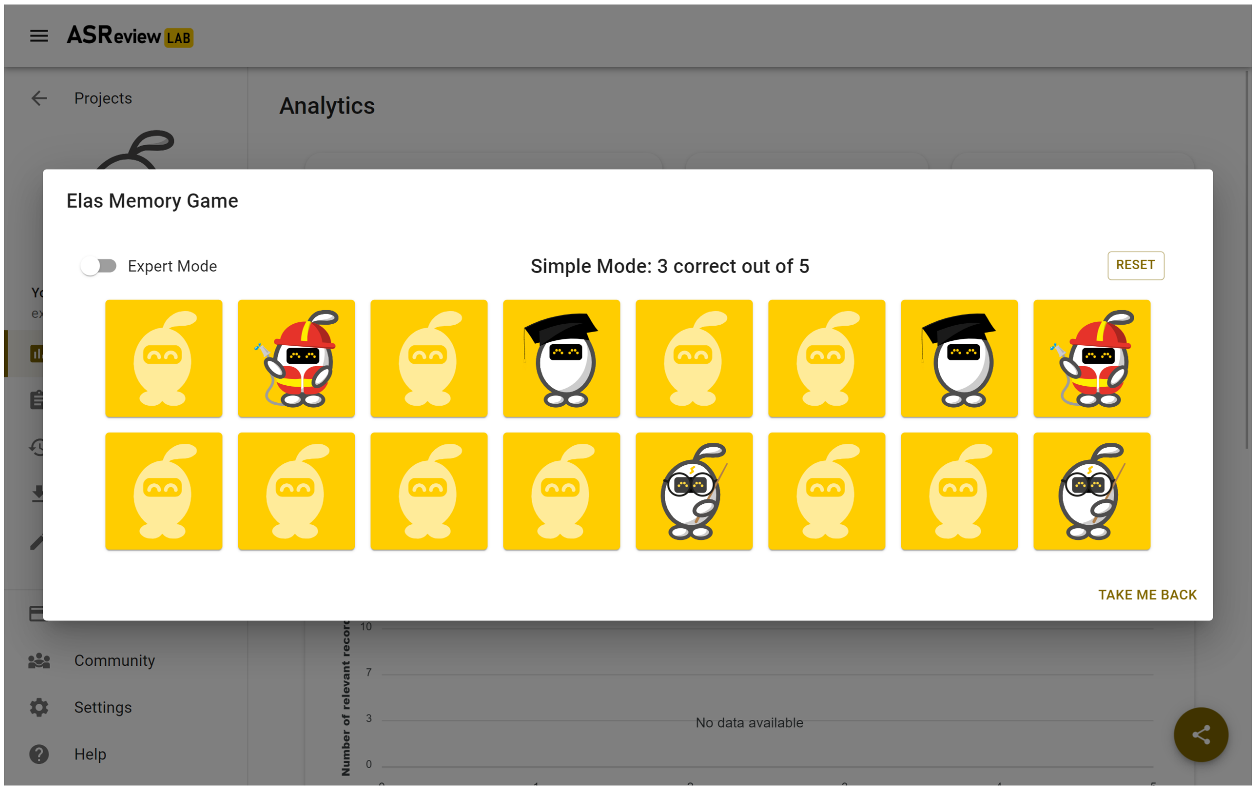
### 12.6.2 Font size

You can make the text on the review screen smaller or larger.

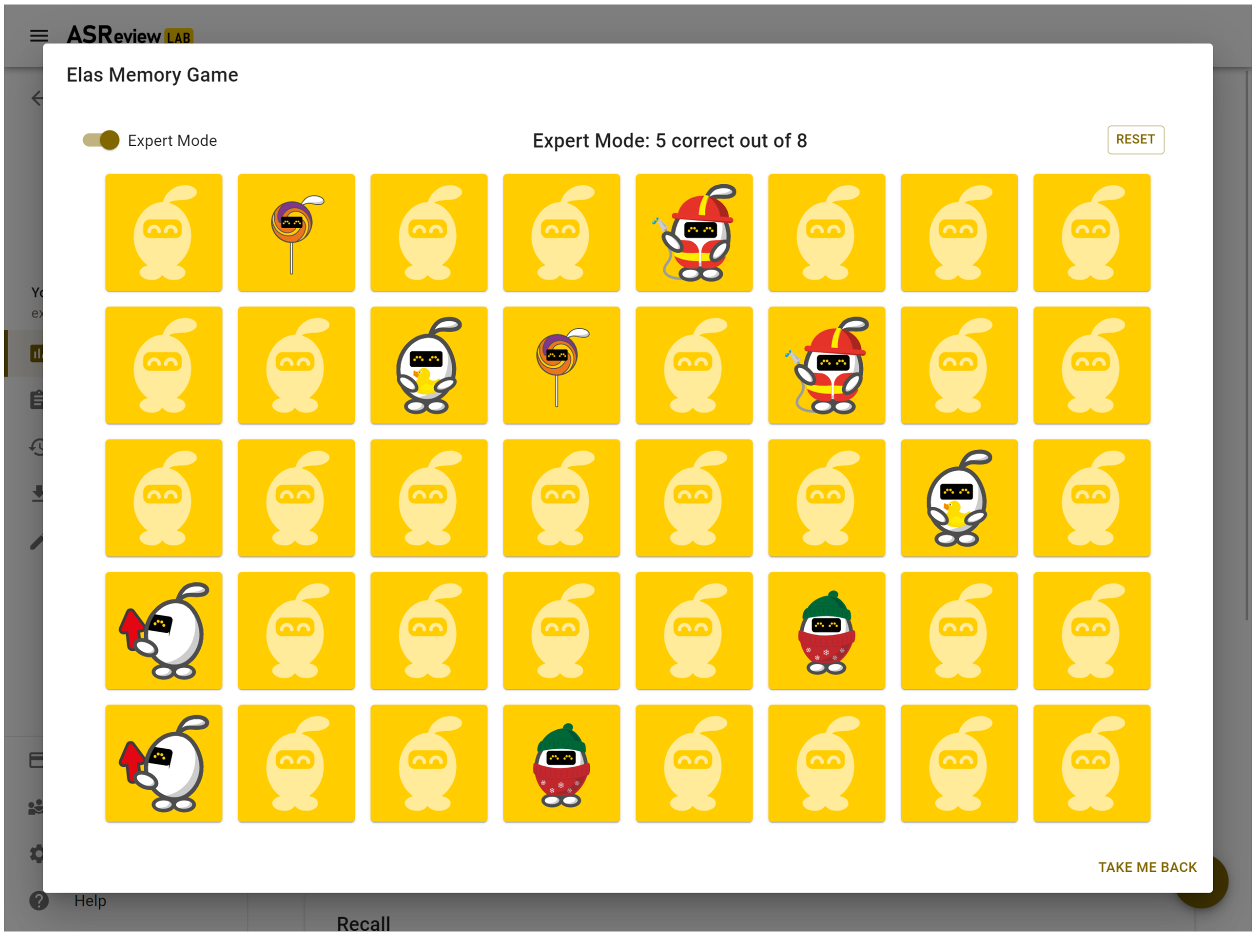
1. *Start ASReview LAB*.
2. Click on *Settings* (bottom left).
3. Go to *Display* and click on *Font size*.
4. Slide the slider to the desired font size.

## 12.7 ELAS Memory Game

If you want a break from screening, you can search for the hidden ELAS memory game.



If you really need a long break, try the expert mode.


ASReview LAB
Elas Memory Game
Expert Mode
Expert Mode: 5 correct out of 8
RESET
TAKE ME BACK
Help
Recall

CHAPTER

THIRTEEN

# PROGRESS AND RESULTS

During screening, you might want to keep track of your progress and to obtain information for your stopping criteria. This section provides documentation on useful tools for these purposes.

## 13.1 Analytics

ASReview LAB offers some insightful statistics, a progress chart, and recall chart to keep track of your screening process and help you to decide when to stop screening.

To open the statistics panel:

1. *Start ASReview LAB*.
2. Open or *Create a project*.
3. Click on Analytics on in the left menu.

### 13.1.1 Summary statistics

The summary statistics are counts of the records in your dataset.

- Total records: the total number of records in your dataset.
- Labeled records: the number of records that you labeled as relevant or irrelevant, including those you added as prior knowledge.
- Relevant records: the number of records that you labeled as relevant, including those you added as prior knowledge.
- Irrelevant records: the number of records that you labeled as irrelevant, including those you added as prior knowledge.
- Irrelevant since last relevant: the number of irrelevant records you have seen since the last relevant record.

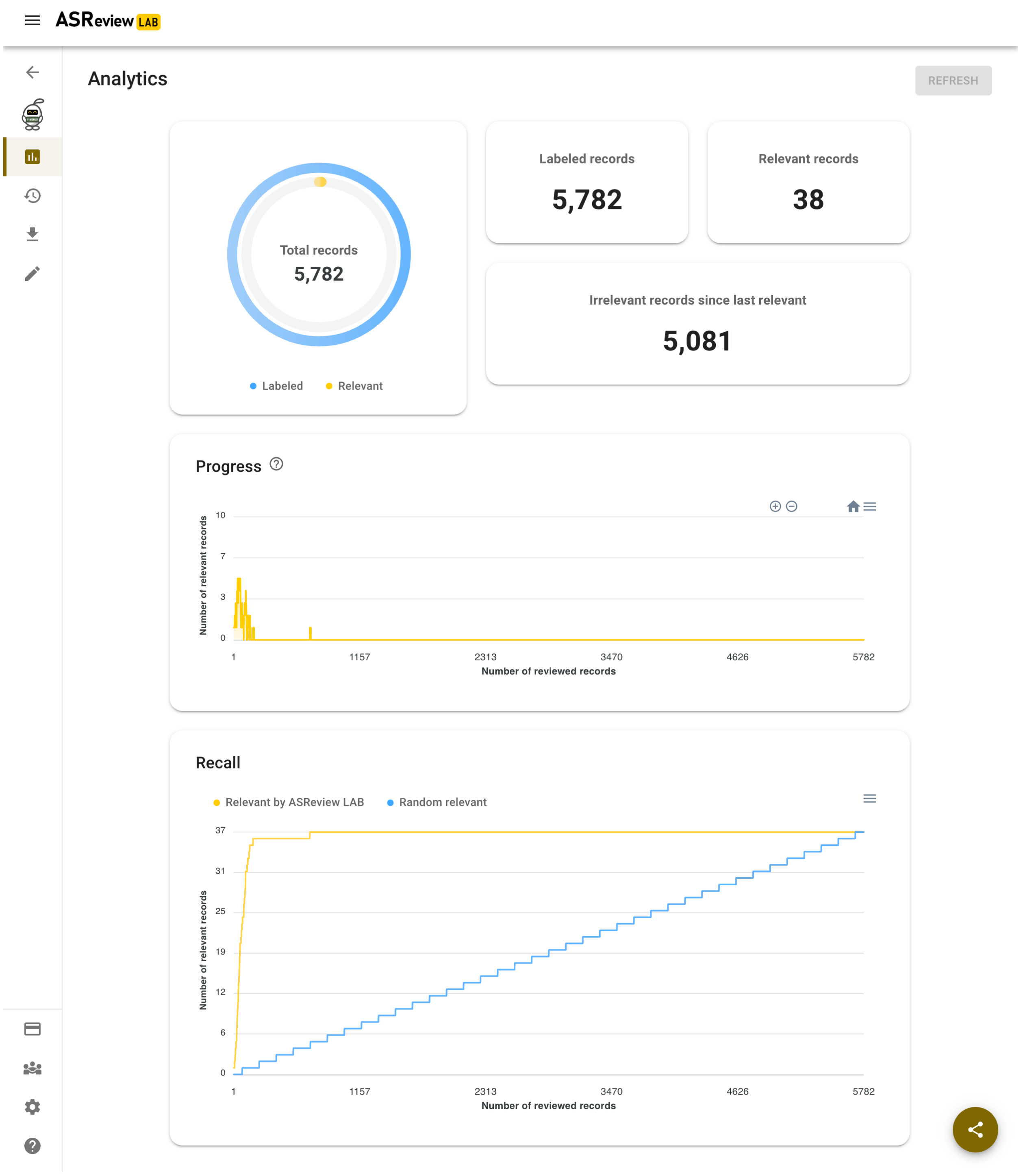


Fig. 1: This figure shows the Analytics page of a fully labeled dataset. All relevant records are found in the first part of the screening.

### 13.1.2 Charts

The charts on the analytics page can be useful to monitor your progress. There is a *Progress* and a *Recall* chart. The charts do not include prior knowledge and are most relevant after you have screened at least 10 records.

**Progress chart**

The progress chart plots the number of relevant records in the last 10 records that you reviewed in ASReview LAB. For example, when you reviewed 100 records, you labeled 3 relevant records between the 91st and 100th reviewed records.

**Recall chart**

The recall chart plots the number of relevant records against the number of records that you reviewed in ASReview LAB. *Relevant by ASReview LAB* refers to the relevant records that you labeled with the assistance of the active learning model. *Random relevant* refers to the relevant records that you might find if you manually reviewed the records so far without the assistance of the active learning model.

**Export Figure**

The plots can be exported as a figure:

1. *Start ASReview LAB*.
2. Open or *Create a project*.
3. Click on Analytics on in the left menu.
4. Click on the hamburger menu next to the Progress or Recall chart.
5. Select *Download SVG* or *PNG* to export a figure, or select *Download CSV* to export the data behind the figure.

## 13.2 Stop screening

The blogpost *ASReview Class 101* and the How to stop screening? discussion provide tips on when to stop with screening.

**Tip:** The number of *irrelevant records since last relevant* will increase the longer you screen.

**Tip:** With *Maximum* as *Query Strategy*, you will see a decline in the number of relevant items in the plots the longer you screen. This may help to decide when to 'stop screening <https://github.com/asreview/asreview/discussions/557>`_.

**Tip:** The data behind the recall plot can be used to calculate the knee-algorithm as a stopping criteria.

## 13.3 Mark project as finished

When you decide to stop screening, you can mark the project as finished. You can undo this at any time. To mark your project as finished:

1. *Start ASReview LAB*.
2. Go to the *Projects dashboard* (http://localhost:5000/projects)
3. Hover the project you want to mark as finished and click on *Options*.
4. Click on *Mark as finished*.

The button to continue screening is now disabled. This can be undone by clicking again on *Mark as in review*.

## 13.4 Export results

You can export the results of your labeling to a RIS, CSV, TSV, or Excel file. A file contains all imported data including your decisions, or a file with a selection of the relevant records only.

The following variables will be added to your tabular dataset:

- The column titled **included** contains the labels as provided by the user: `0` = not relevant, `1` = relevant and if missing it means the record is not seen during the screening process.
- The column titled **asreview_ranking** contains an identifier to preserve the rank ordering as described below.
- The column **ASReview_prior** contains a label `1` if a record has been used to train the first iteration of the model, a label `0` if not used for training, and empty when the record was not seen.
- The column **asreview_label_to_validate** is added in the exploration mode and contains the labels initially present in the data.
- The column **Notes** contain any notes you made during screening.

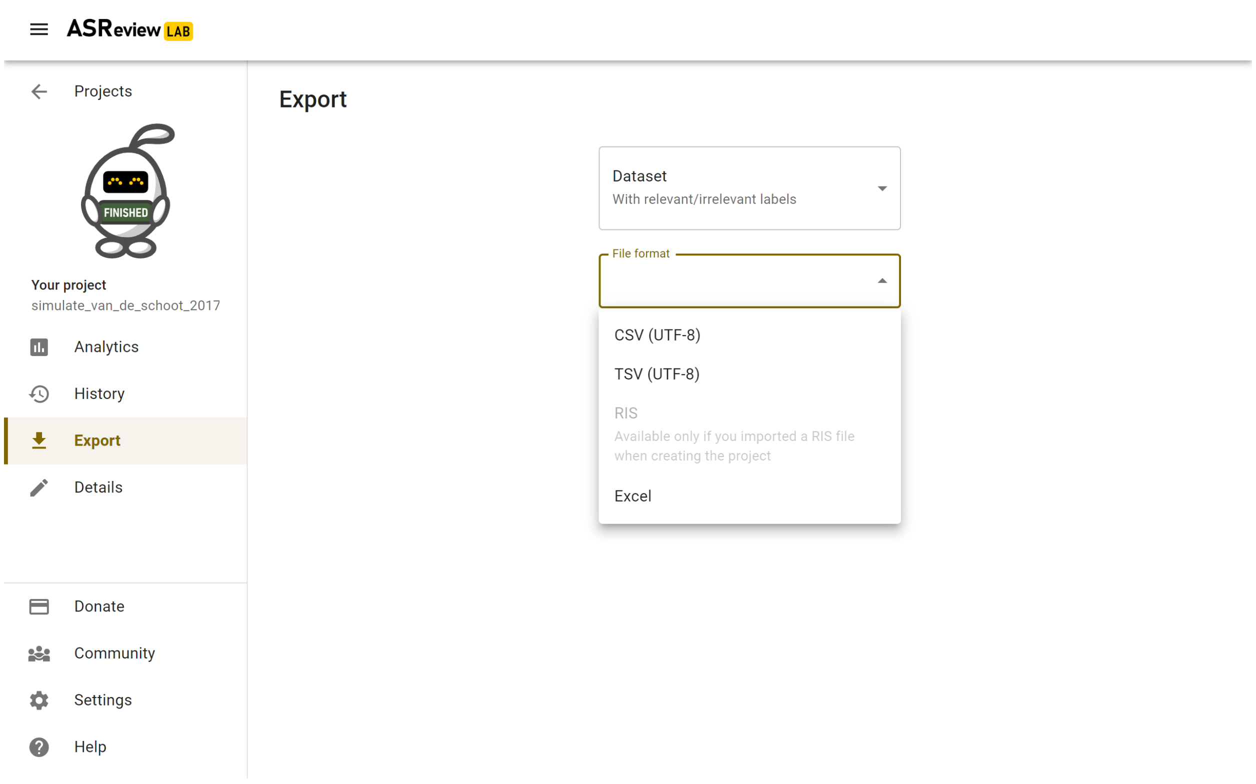
For RIS files, the labels **ASReview_relevant**, **ASReview_irrelevant**, **ASReview_not_seen**, and **ASReview_prior**, **ASReview_validate_relevant/irrelevant/not_seen** are stored with the *N1* (Notes) tag. In citation managers like Zotero and Endnote, the labels can be used for making selections.

The file is ordered as follows:

1. All relevant records you have seen in the order they were shown during the screening process.
2. All records not seen during the screening and ordered from most to least relevant according to the last iteration of the model.
3. All non-relevant records are presented in the order these are shown during the screening process.

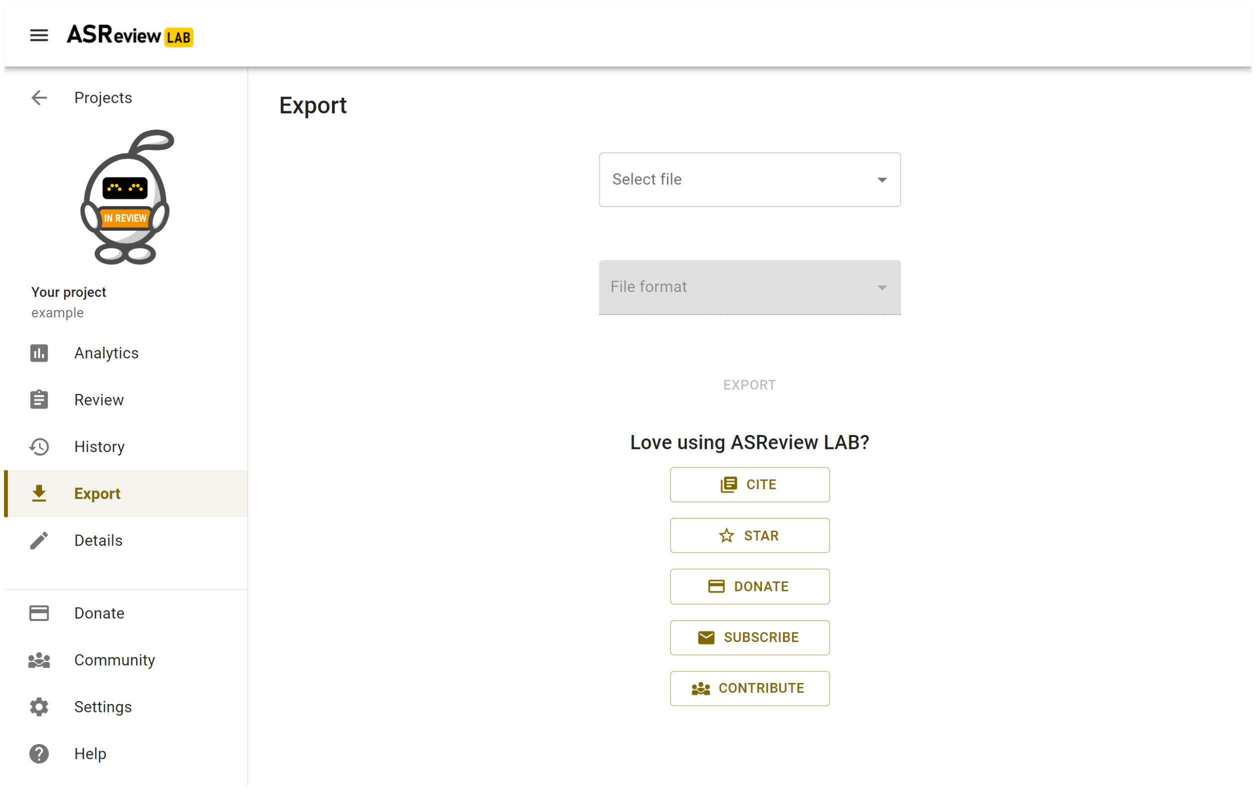
To download your results follow these steps:

1. *Start ASReview LAB*.
2. Open or *Create a project*.
3. Click on *Export* in the menu on the left.
4. Select *Dataset*.
5. Select the file type for prefer: i.e. Excel, RIS, TSV, or CSV file.
6. Save the file to your device.



**Note:** A RIS file can only be exported if a RIS file is imported.

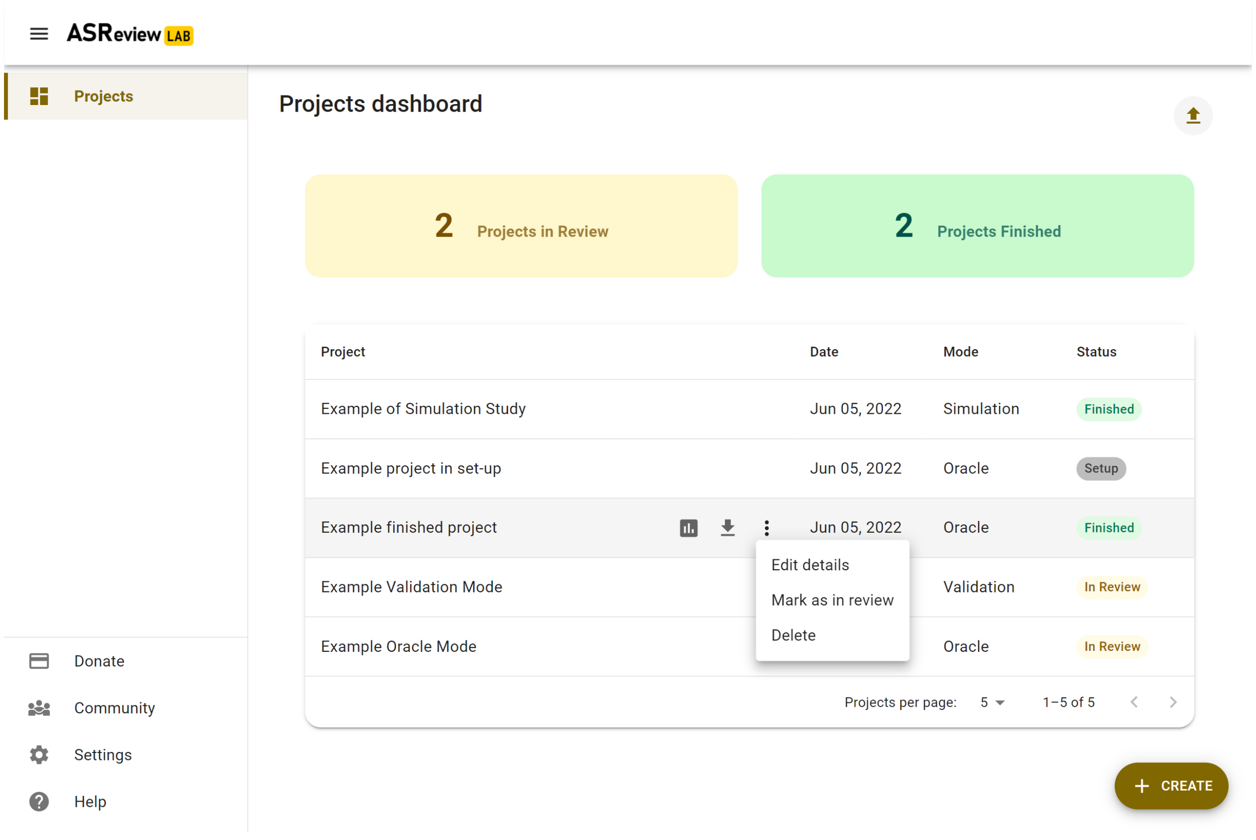
Love using ASReview? On the Export screen you can get inspired how you can give back to our open-source and community-driven project.


ASReview LAB
Projects
IN REVIEW
Your project
example
Analytics
Review
History
Export
Details
Donate
Community
Settings
Help
Export
Select file
File format
EXPORT
Love using ASReview LAB?
CITE
STAR
DONATE
SUBSCRIBE
CONTRIBUTE

# CHAPTER FOURTEEN

# MANAGE PROJECTS

ASReview LAB offers the options to import and export projects. This can be useful for sharing results, archiving projects, and for backup purposes.



| Project | Date | Mode | Status |
|---|---|---|---|
| Example of Simulation Study | Jun 05, 2022 | Simulation | Finished |
| Example project in set-up | Jun 05, 2022 | Oracle | Setup |
| Example finished project | Jun 05, 2022 | Oracle | Finished |
| Example Validation Mode | | Validation | In Review |
| Example Oracle Mode | | Oracle | In Review |

## 14.1 Import Project

To import a project:

1. *Start ASReview LAB*.
2. Go to the *Projects dashboard* (http://localhost:5000/projects)
3. Click on the *Import project* icon on the top right.
4. Click on *Select file* and select a project from your device (with `.asreview` extension.

5. Open the project from the *Projects dashboard*.

## 14.2 Export Project

The ASReview project file (extension `.asreview`) can be exported from ASReview LAB. The file contains the dataset, review history, notes, and model configuration. It can be imported into ASReview LAB on a different device, which allows other users to replicate the project, or continue the systematic review.

To export your project:

1. *Start ASReview LAB*.
2. Go to the *Projects dashboard* (http://localhost:5000/projects)
3. Hover the project you want to export and click on the *Export* icon.
4. Click on *Select file* and click on *Project*.
5. Click on *Export*

You will be asked where to save the ASReview file (extension *.asreview*).

## 14.3 Delete Project

To permanently delete a project, including ALL files:

1. *Start ASReview LAB*.
2. Go to the *Projects dashboard* (http://localhost:5000/projects)
3. Hover the project you want to export and click on *Options*.
4. Click on *Delete forever*.
5. This action cannot be made undone, ASReview LAB will ask you to confirm by typing in the project title.

CHAPTER

FIFTEEN

# OVERVIEW

ASReview LAB offers three different solutions to run simulations with the:

- *Webapp (the frontend)*
- *Command line interface*
- *Python API*

## 15.1 What is a simulation?

A simulation involves mimicking the screening process with a certain model. As it is already known which records are labeled as relevant, the software can automatically reenact the screening process as if a human was labeling the records in interaction with the Active Learning model.

## 15.2 Why run a simulation?

Simulating with ASReview LAB has multiple purposes. First, the performance of one or multiple models can be measured by different metrics (see *Analyzing results*). A convenient one is that you can investigate the amount of work you could have saved by using active learning compared to your manual screening process.

Suppose you don't know which model to choose for a new (unlabeled) dataset. In that case, you can experiment with the best performing combination of the classifier, feature extraction, query strategy, and balancing and test the performance on a labeled dataset with similar characteristics.

You could also use the simulation mode to benchmark your own model against existing models for different available datasets. ASReview LAB allows for adding new models via a template.

You can also find 'odd' relevant records in a 'classical' search. Such records are typically found isolated from most other records and might be worth closer inspection

## 15.3 Datasets for simulation

Simulations require *fully labeled datasets* (labels: `0` = irrelevant, `1` = relevant). Such a dataset can be the result of an earlier study. ASReview offers also fully labeled datasets via the SYNERGY dataset. These datasets are available via the user interface in the *Data* step of the setup and in the command line with the prefix *synergy:* (e.g. *synergy:van_de_schoot_2018*).

**Tip:** When you import your data, make sure to remove duplicates and to retrieve as many abstracts as possible (See Importance-of-abstracts blog for help). With clean data you benefit most from what *active learning* has to offer.

## 15.4 Cloud environments

For advanced scenarios, such as executing ASReview simulations in cloud environments or running them in parallel, consult the specialized cloud usage guide. This guide provides tailored instructions for a variety of use cases, including simulations on cloud platforms such as SURF, Digital Ocean, AWS, Azure, and leveraging Kubernetes for large-scale simulation tasks. More information can be found in the paper: Optimizing ASReview simulations: A generic multi-processing solution for 'light-data' and 'heavy-data' users

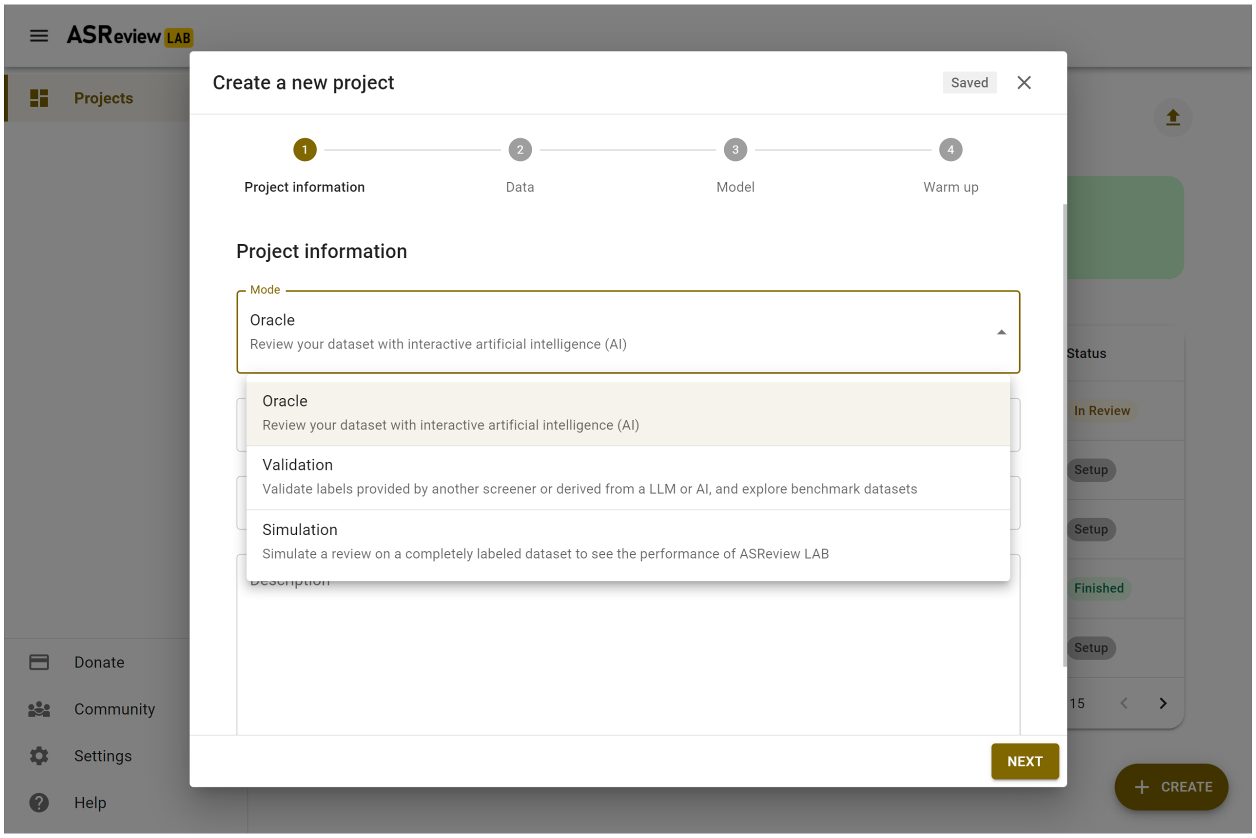
CHAPTER

SIXTEEN

# SIMULATE VIA THE WEBAPP

To run a simulation in the ASReview webapp, create a project as described in *Create a project*. Most of the steps of the setup are identical or straightforward. In this section, some of the differences are highlighted.

In the step on *Project Information*, select the “Simulation” mode (see figure below).



Project information

Mode
Oracle
Review your dataset with interactive artificial intelligence (AI)

Oracle
Review your dataset with interactive artificial intelligence (AI)

Validation
Validate labels provided by another screener or derived from a LLM or AI, and explore benchmark datasets

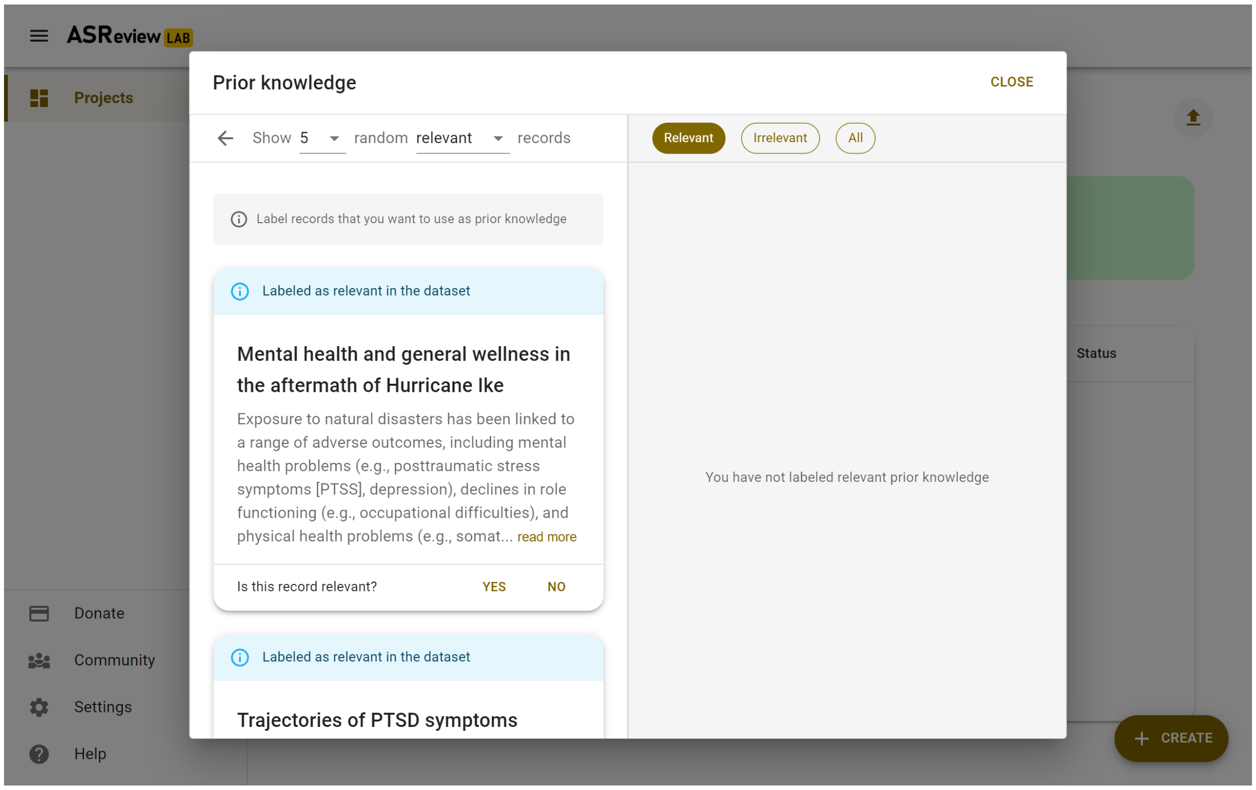
Simulation
Simulate a review on a completely labeled dataset to see the performance of ASReview LAB

In the step *Data*, import a *fully labeled dataset* or use one of the benchmark datasets.

Selecting prior knowledge is relatively easy. In case you know relevant records to start with, use the search function. In case you don’t, use the *Random* option. Toggle the button “Relevant” on top to see some random irrelevant records. Label some relevant and some irrelevant records.

The step *Warm up* is differs slightly from the Oracle and Validation mode. This step starts the simulation, after some seconds, it will return “Got it”. This means, the simulation runs further in the background. You are returned to the Analytics page.

This page now has a refresh button on the top right. If the simulation is not finished yet, you can refresh the page or



ASReview LAB

Projects

Prior knowledge CLOSE

Show 5 random relevant records

Relevant Irrelevant All

Label records that you want to use as prior knowledge

Labeled as relevant in the dataset

Mental health and general wellness in the aftermath of Hurricane Ike

Exposure to natural disasters has been linked to a range of adverse outcomes, including mental health problems (e.g., posttraumatic stress symptoms [PTSS], depression), declines in role functioning (e.g., occupational difficulties), and physical health problems (e.g., somat... read more

Is this record relevant? YES NO

Labeled as relevant in the dataset

Trajectories of PTSD symptoms

You have not labeled relevant prior knowledge

Status

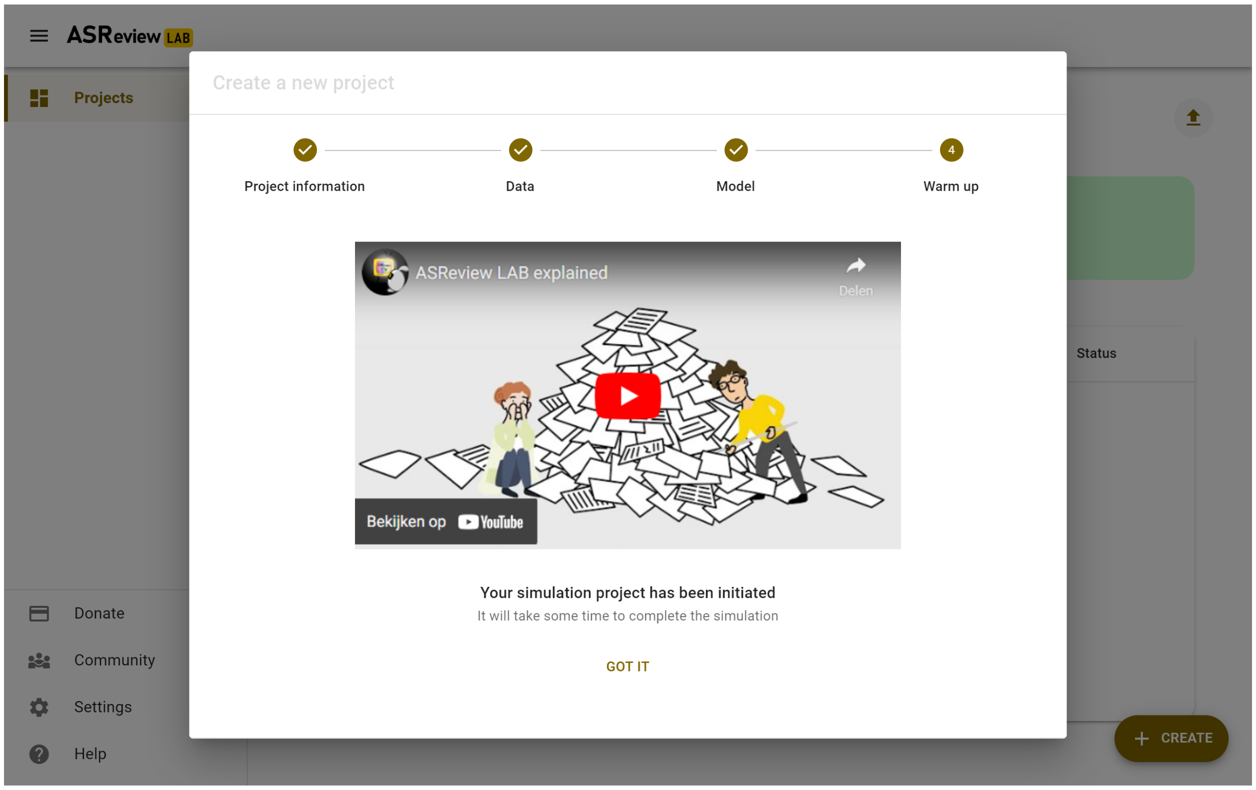
Donate

Community

Settings

Help

CREATE



ASReview LAB

Projects

Create a new project

Project information

Data

Model

4 Warm up

Your simulation project has been initiated

It will take some time to complete the simulation

GOT IT

Status

Donate

Community

Settings

Help

CREATE

use the refresh button to follow the progress. After a while, the Elas mascot on the left will hold a sign with “finished”. Your simulation is now finished and you can study the results in the analytics page.

CHAPTER

# SEVENTEEN

# SIMULATION VIA COMMAND LINE

ASReview LAB comes with a command line interface for simulating the performance of ASReview algorithm.

## 17.1 Getting started

The simulation command line tool can be accessed directly like:

```
asreview simulate MY_DATASET.csv -s MY_SIMULATION.asreview
```

This performs a simulation with the default active learning model, where `MY_DATASET.csv` is the path to the *Fully labeled data* you want to simulate. The result of the simulation is stored, after a successful simulation, at `MY_SIMULATION.asreview` where `MY_SIMULATION` is the filename you prefer and the extension is `.asreview` (ASReview project file extension).

## 17.2 Simulation progress

The progress of the simulation is given with two progress bars. The top one is used to count the number of relevant records found. The bottom one monitors the number of records labeled. By default (with `--stop-if min`), the simulation stops once the the top progress bar reaches 100%.

```
Simulation started

Relevant records found: 100%|| 43/43 [00:03<00:00, 13.42it/s]
Records labeled       :   7%|                                | 420/6189 [00:03
↪<00:43, 133.58it/s]

Simulation finished
```

## 17.3 Command line arguments for simulating

The command `asreview simulate --help` provides an overview of available arguments for the simulation.

Each of the sections below describe the available arguments. The example below shows how you can set the command line arguments. This can be helpful if you are new to the using the command line. For example, you want to change the query strategy being used. The command line and this documentation show `-q, --query_strategy QUERY_STRATEGY`. The default is `max`. If you want to change it to `max_random`, you use:

```
asreview simulate MY_DATASET.csv -s MY_SIMULATION.asreview -q max_random
```

### 17.3.1 Dataset

**dataset**

Required. File path or URL to the dataset or one of the SYNERGY datasets.

You can also use one of the *SYNERGY dataset*. Use the following command and replace `DATASET_ID` by the dataset ID.

```
asreview simulate synergy:DATASET_ID
```

For example:

```
asreview simulate synergy:van_de_schoot_2018 -s myreview.asreview
```

### 17.3.2 Active learning

**-e, --feature_extraction** `FEATURE_EXTRACTION`

The default is TF-IDF (`tfidf`). More options and details are listed in *asreview.models.feature_extraction*.

**-m, --model** `MODEL`

The default is Naive Bayes (`nb`). More options and details are listed in *asreview.models.classifiers*.

**-q, --query_strategy** `QUERY_STRATEGY`

The default is Maximum (`max`). More options and details are listed in *asreview.models.query*.

**-b, --balance_strategy** `BALANCE_STRATEGY`

The default is `double`. The balancing strategy is used to deal with the sparsity of relevant records. More options and details are listed in *asreview.models.balance*

**--seed** `SEED`

To make your simulations reproducible you can use the `--seed` and `--init_seed` options. 'init_seed' controls the starting set of papers to train the model on, while the 'seed' controls the seed of the random number generation that is used after initialization.

**--embedding** `EMBEDDING_FP`

File path of embedding matrix. Required for LSTM models.

### 17.3.3 Prior knowledge

By default, the model initializes with one relevant and one irrelevant record. You can set the number of priors by `--n_prior_included` and `--n_prior_excluded`. However, if you want to initialize your model with a specific set of starting papers, you can use `--prior_idx` to select the indices of the papers you want to start the simulation with. When no prior knowledge is assigned (using `--n_prior_included 0 --n_prior_excluded 0`), the first records from the dataset are employed as priors in the order they were provided until the first 0 and 1 are encountered.

The following options can be used to label prior knowledge:

**--n_prior_included** `N_PRIOR_INCLUDED`

The number of prior included papers. Only used when `prior_idx` is not given. Default 1.

**--n_prior_excluded** `N_PRIOR_EXCLUDED`
: The number of prior excluded papers. Only used when `prior_idx` is not given. Default 1.

**--prior_idx** `[PRIOR_IDX [PRIOR_IDX ...]]`
: Prior indices by rownumber (rownumbers start at 0).

**--init_seed** `INIT_SEED`
: Seed for setting the prior indices if the prior_idx option is not used. If the option prior_idx is used with one or more index, this option is ignored.

### 17.3.4 Simulation setup

**--n_instances** `N_INSTANCES`
: Controls the number of records to be labeled before the model is retrained. Increase `n_instances`, for example, to reduce the time it takes to simulate. Default 1.

**--stop_if** `STOP_IF`
: The number of label actions to simulate. Default, 'min' will stop simulating when all relevant records are found. Use -1 to simulate all labels actions.

### 17.3.5 Save

**--state_file** `STATE_FILE`, **-s** `STATE_FILE`
: Location to ASReview project file of simulation.

## 17.4 Algorithms

The command line interface provides an easy way to get an overview of all available active learning model elements (classifiers, query strategies, balance strategies, and feature extraction algorithms) and their names for command line usage in ASReview LAB. It also includes models added via *Extensions*. The following command lists the available models:

```
asreview algorithms
```

See *Extensions* for more information on developing new models and install them via extensions.

Some models require additional dependencies to be installed. Use `pip install asreview[all]` to install all additional dependencies at once or check the installation instruction in section *Models* of the *API Reference*.

CHAPTER

EIGHTEEN

# ANALYZING RESULTS

After a simulation, the results are stored in the ASReview project file (extension *.asreview*). This file contains a large number of variables and logs on the simulation. The data can be extracted from the project file via the API or with one of the available extensions. See *these examples on the Project API* for more information about opening the project file.

One readily available extension for analyzing the results of a simulation is ASReview Insights. This extension offers valuable tools for plotting the recall and extracting the statistical results of several performance metrics, such as the Work Saved over Sampling (WSS), the proportion of Relevant Record Found (RRF), the Extra Relevant records Found (ERF), and the Average Time to Discover (ATD).

Install ASReview Insights directly from PyPi:

```
pip install asreview-insights
```

Detailed documentation on the extension can be found on the ASReview Insights project page.

CHAPTER

NINETEEN

# SIMULATE WITH PYTHON API

The API is still under development and can change at any time without warning.

For more control over the workings of the ASReview software, the ASReview Python API can be used. For example, it is possible to use custom models or implement different sampling strategies. This example shows how to simulate a review with the ASReview API and store the results in an ASReview project file.

Please keep in mind that the ASReview API is experimental at the moment. Improvements and simplifications are planned.

[1]:
```
from pathlib import Path

from asreview import ASReviewData, ASReviewProject
from asreview.review import ReviewSimulate
```

Create a temporary folder for the results and examples in this document.

[2]:
```
project_path = Path("tmp_data")
project_path.mkdir(exist_ok=True)
```

Create an *ASReviewProject* to store the results

[3]:
```
# Create a project object and folder
project = ASReviewProject.create(
    project_path=project_path / "api_simulation",
    project_id="api_example",
    project_mode="simulate",
    project_name="api_example",
)
```

Add a dataset to the project folder in the folder `data` (can also be stored somewhere else, but it is advised to use the data folder). In the following example, a dataset is downloaded from the benchmark platform with CURL (macOS, Unix systems).

[4]:
```
%%bash
curl https://raw.githubusercontent.com/asreview/systematic-review-datasets/metadata-v1-
↪final/datasets/van_de_Schoot_2017/output/van_de_Schoot_2017.csv > tmp_data/api_
```

(continues on next page)

(continued from previous page)

```
↪simulation/data/van_de_Schoot_2017.csv
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  9.9M  100  9.9M    0     0  13.2M      0 --:--:-- --:--:-- --:--:-- 13.2M
```

Add the reference to the dataset to the project.

```
[5]: project.add_dataset("van_de_Schoot_2017.csv")
```

Setup the models.

```
[6]: from asreview.models.classifiers import NaiveBayesClassifier
     from asreview.models.query import MaxQuery
     from asreview.models.balance import DoubleBalance
     from asreview.models.feature_extraction import Tfidf

     # Select models to use
     train_model = NaiveBayesClassifier()
     query_model = MaxQuery()
     balance_model = DoubleBalance()
     feature_model = Tfidf()
```

Run the simulation with the `ReviewSimulate` class.

```
[7]: data_obj = ASReviewData.from_file(
         Path("tmp_data", "api_simulation", "data", "van_de_Schoot_2017.csv")
     )
```

```
[8]: # Initialize the simulation reviewer
     reviewer = ReviewSimulate(
         as_data=data_obj,
         model=train_model,
         query_model=query_model,
         balance_model=balance_model,
         feature_model=feature_model,
         n_instances=10,
         project=project,
         n_prior_included=1,
         n_prior_excluded=1,
     )
```

```
[9]: # Start the review process
     project.update_review(status="review")
     try:
         reviewer.review()
         project.mark_review_finished()
```

(continues on next page)

(continued from previous page)

```
except Exception as err:
    project.update_review(status="error")
    raise err
```

Export the project to a location of choice, in this case `tmp_data/api_example.asreview`.

[10]:
```
# Finish and export the project
project.export(Path("tmp_data", "api_example.asreview"))
```

The following code removes the temporary folder that was created:

[11]:
```
import shutil

shutil.rmtree(project_path)
```

CHAPTER

TWENTY

# ASREVIEW LAB SERVER

ASReview LAB Server is a self-hosted, secure version of ASReview LAB. It is designed for facilitate users who want to use ASReview LAB but without the need to install it on their own computer. The web application that can be accessed from any device with a web browser and can be used on desktops, laptops, tablets, and mobile devices. ASReview LAB Server enables users to create an account or connect via their GitHub, ORCID, or Google accounts.

See the *Server configuration* details for more information on how to configure your ASReview LAB on your server.

## 20.1 Features

ASReview LAB provides two options for creating an account: by connecting with your GitHub, ORCID, or Google account, or by creating an account. All information is stored securely on the ASReview LAB server and fully self-hosted.

### 20.1.1 Log in with GitHub, ORCID, or Google

ASReview LAB Server provides a easy way to log in with your GitHub, ORCID, or Google account.

See the *Server configuration* details for more information on how to configure your ASReview on your server to enable this feature.

### 20.1.2 Create account

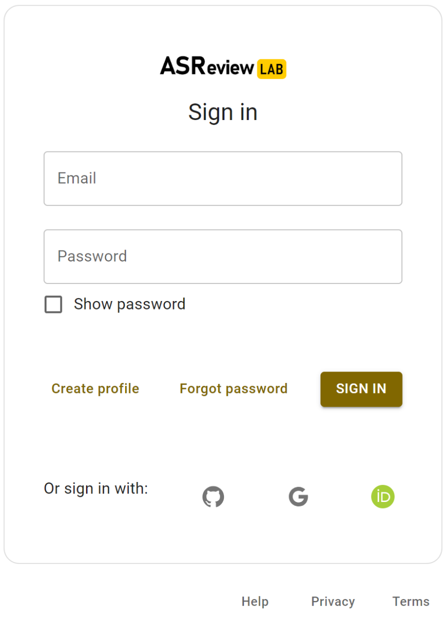
ASReview LAB Server provides a easy way to create an account with your email.

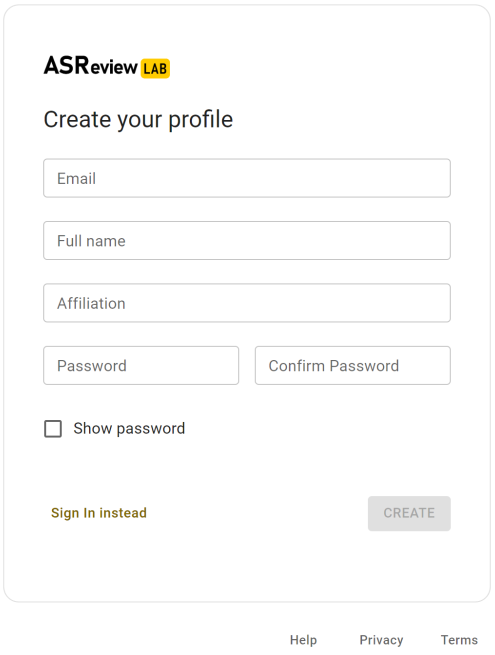
## 20.2 Installation

ASReview LAB server is installed in the same way as ASReview LAB. See the *Installation* instructions for more information. See *Server configuration* for more information on how to configure authentication on your ASReview LAB server.

Optional: If you want to make use of the PostgreSQL database, you need to install the *psycopg2* package. This can be done by running the following command:

```
pip install psycopg2
```


ASReview LAB
Sign in
Email
Password
Show password
Create profile
Forgot password
SIGN IN
Or sign in with:
Help
Privacy
Terms


ASReview LAB
Create your profile
Email
Full name
Affiliation
Password
Confirm Password
Show password
Sign In instead
CREATE
Help
Privacy
Terms

# CHAPTER TWENTYONE

# SERVER CONFIGURATION

ASReview LAB offers a number of options to run the application on a server. It is possible to run the application on a server with or without authentication. The latter is the default option. This page describes how to configure the ASReview LAB application to run on a server with authentication enabled. With authentication enabled, users can to run their projects in their own separate workspaces. Authentication requires the storage of user accounts and link these accounts to projects. Currently we are using a SQLite database (asreview.development.sqlite or asreview.production.sqlite) in the ASReview projects folder to store that information.

## 21.1 Bare bones authentication

The most basic configuration of the ASReview application with authentication is to run the application from the CLI with the `--enable-auth` flag. The application will start with authentication enabled and will create a SQLite database if it does not exist. The database will be stored in the ASReview projects folder. The database contains the user accounts and links them to projects.

Start the application with authentication enabled:

```
asreview lab --enable-auth --secret-key=<secret key> --salt=<salt>
```

where `--enable-auth` forces the application to run in an authenticated mode, `<secret key>` is a string that is used for encrypting cookies and `<salt>` is a string that is used to hash passwords. The `--secret-key` and `--salt` parameters are mandatory if authentication is required.

To create user accounts, one can use the `add-users` command of the `auth-tool` sub command of the ASReview application:

```
asreview auth-tool add-users --db-uri=sqlite:////path/example.sqlite
```

For more information about auth-tool and creating users, see the section *Create user accounts* below.

## 21.2 Full authentication configuration

To configure the authentication in more detail we need to create a TOML file that contains all relevant authentication parameters. The parameters in that TOML file will override parameters that were passed in the CLI. Below is an example of a TOML file (extension *.toml*) that enables authentication and OAuth with Github, Orcid and Google. It also enables email verification and allows users to create their own accounts. The email server is configured to confirm new accounts and to allow users to retrieve a new password if they forget it. The TOML file also contains the necessary parameters to run the application in a secure way (https).

```
DISABLE_LOGIN = false
SECRET_KEY = "<secret key>"
SECURITY_PASSWORD_SALT = "<salt>"
SESSION_COOKIE_SECURE = true
REMEMBER_COOKIE_SECURE = true
SESSION_COOKIE_SAMESITE = "Lax"
SQLALCHEMY_TRACK_MODIFICATIONS = true
ALLOW_ACCOUNT_CREATION = true
ALLOW_TEAMS = false
EMAIL_VERIFICATION = false

MAIL_SERVER = "<smtp-server>"
MAIL_PORT = 465
MAIL_USERNAME = "<smtp-server-username>"
MAIL_PASSWORD = "<smtp-server-password>"
MAIL_USE_TLS = false
MAIL_USE_SSL = true
MAIL_DEFAULT_SENDER = "<preferred reply email address>"

[OAUTH]
        [OAUTH.GitHub]
        AUTHORIZATION_URL = "https://github.com/login/oauth/authorize"
        TOKEN_URL = "https://github.com/login/oauth/access_token"
        CLIENT_ID = "<GitHub client ID>"
        CLIENT_SECRET = "<GitHub client secret>"
        SCOPE = ""

        [OAUTH.Orcid]
        AUTHORIZATION_URL = "https://sandbox.orcid.org/oauth/authorize"
        TOKEN_URL = "https://sandbox.orcid.org/oauth/token"
        CLIENT_ID = "<Orcid client ID>"
        CLIENT_SECRET = "<Orcid client secret>"
        SCOPE = "/authenticate"

        [OAUTH.Google]
        AUTHORIZATION_URL = "https://accounts.google.com/o/oauth2/auth"
        TOKEN_URL = "https://oauth2.googleapis.com/token"
        CLIENT_ID = "<Google client ID>"
        CLIENT_SECRET = "<Google client secret>"
        SCOPE = "profile email"
```

Store the TOML file on the server and start the ASReview application from the CLI with the `--flask-configfile` parameter:

```
asreview lab --flask-configfile=<path-to-TOML-config-file>
```

A number of the keys in the TOML file are standard Flask parameters. The keys that are specific for authenticating ASReview are summarized below:

- DISABLE_LOGIN: if set to `false` the application will start with authentication. If the SQLite database does not exist, one will be created during startup.
- SECRET_KEY: the secret key is a string that is used to encrypt cookies and is mandatory if authentication is required.

- SECURITY_PASSWORD_SALT: another string used to hash passwords, also mandatory if authentication is required.
- SESSION_COOKIE_SAMESITE: Restrict how cookies are sent with requests from external sites. In the example the value is set to "Lax" which is the recommended option. If backend and frontend are served on different domains set to the string "None".
- ALLOW_ACCOUNT_CREATION: enables account creation by users, either by front- or backend.
- EMAIL_VERIFICATION: used in conjunction with ALLOW_ACCOUNT_CREATION. If set to `true` the system sends a verification email after account creation. Only relevant if the account is __not__ created by OAuth. This parameter can be omitted if you don't want verification.
- MAIL_<PAR>: configuration parameters to setup the SMTP email server that is used for email verification. It also allows users to retrieve a new password after forgetting it. Don't forget to enter the reply address (MAIL_DEFAULT_SENDER) of your system emails. Remove these parameters if system emails for verification and password retrieval are unwanted.
- OAUTH: an authenticated ASReview application may integrate with the OAuth functionality of Github, Orcid and Google. Provide the necessary OAuth login credentails (for Github, Orcid en Google). Please note that the *AUTHORIZATION_URL* and *TOKEN_URL* of the Orcid entry are sandbox-urls, and thus not to be used in production. Omit this parameter if OAuth is unwanted.

The `SQLALCHEMY_DATABASE_URI` key is not included in the TOML file. This key is used to configure the database connection. The default value is `sqlite:///asreview.production.sqlite`. This means that the application will use the SQLite database in the ASReview projects folder. If you would like to use a different database, you can add the `SQLALCHEMY_DATABASE_URI` key to the TOML file.

Set the `SQLALCHEMY_DATABASE_URI` environment variable to the path of the database. For example, to use the SQLite database in the ASReview projects folder:

```
FLASK_SQLALCHEMY_DATABASE_URI = "sqlite:///asreview.production.sqlite"
```

## 21.3 PostgreSQL database

You can replace the SQLite database with a PostgreSQL database. This requires an extra step during installation and an extra step in the configuration file:

1. Install the psycopg2 package. At the time of this writing 2 versions of this package exist: `psycopg2` and `psycopg2-binary`. According to the documentation the binary version works on most operating systems.
2. Then add the `SQLALCHEMY_DATABASE_URI` key to the config file:

```
SQLALCHEMY_DATABASE_URI = "postgresql+psycopg2://username:password@host:port/database_
↪name"
```

Create authentication database and tables with auth-tool

Server administrators can create a database for authentication with the `auth-tool` sub command of the ASReview application:

```
asreview auth-tool create-db --db-uri=sqlite:////path/example.sqlite
```

Please note that in this example, the –db-uri option is explicitly configured. However, it is not mandatory. If access to the authentication database is needed, the auth-tool utility first checks whether the –db-uri option has been provided. If not, it then examines the presence of the SQLALCHEMY_DATABASE_URI environment variable. In the absence of this variable as well, the script defaults to utilizing the database URI associated with the standard SQLite database pre-configured in the ASReview folder.

## 21.4 Create user accounts with auth-tool

Create user accounts interactively or by using a JSON string to bulk insert the accounts with `add-users`. To add user accounts interactively run the following command:

```
asreview auth-tool add-users --db-uri=sqlite:////path/example.sqlite
```

The tool will prompt you if you would like to add a user account. Type `Y` to continue and enter an email address, name, affiliation (not required) and a password for every person. Continue to add as many users as you would like.

If you would like to bulk insert user accounts use the `--json` option:

```
asreview auth-tool add-users \
        --db-uri=sqlite:////path/example.sqlite \
        -j "[{\"email\": \"name@email.org\", \"name\": \"Name of User\", \"affiliation\":
↪ \"Some Place\", \"password\": \"1234@ABcd\"}]"
```

The JSON string represents a Python list with a dictionary for every user account with the following keys: `email`, `name`, `affiliation` and `password`. Note that passwords require at least one symbol. These symbols, such as the exclamation mark, may compromise the integrity of the JSON string.

## 21.5 List projects with auth-tool

The `auth-tool` sub command of the ASReview application can be used to list projects.

Lists all projects with the `list-projects` command:

```
asreview auth-tool list-projects
```

List the projects in JSON format with the `--json` flag:

```
asreview auth-tool list-projects --json
```

The command returns a convenient JSON string that can be used to bulk insert and link projects into the database. The string represents a list containing a dictionary for every project.

## 21.6 List users with auth-tool

The `auth-tool` sub command of the ASReview application can be used to list users.

Lists all users with the `list-users` command:

```
asreview auth-tool list-users
```

## 21.7 Migrate projects from unauthenticated to authenticated

By default, the ASReview application runs in an unauthenticated mode. This means that all projects are stored in the same workspace. This is fine for a single user, but not for multiple users. If you would like to run the application in an authenticated mode, you need to convert the existing projects into authenticated ones with user identifiers assigned to each project. If you don't do this, you won't see any projects in the authenticated mode.

First, list all users with the `list-users` command. Create users if you don't have users yet.

```
asreview auth-tool list-users --db-uri=sqlite:////path/example.sqlite
```

List all projects with the `list-projects` command. The command returns a

```
asreview auth-tool list-projects
```

Migrate the projects into the authenticated database can be done interactively:

```
asreview auth-tool link-projects --db-uri=sqlite:////path/example.sqlite
```

The tool will list project by project and asks what the ID of the owner is. That ID can be found in the user list below the project information.

You can also insert all project information by using the JSON string that was produced with the `list-projects` command. Add user identifiers to each project in the JSON string. For example, if the user ID of the owner is `15`, the JSON string should look like this

```
asreview auth-tool link-projects \
        --db-uri=sqlite:////path/example.sqlite \
        --json "[{\"folder\": \"project-id\", \"version\": \"1.3\", \"project_id\": \
↪"project-id\", \"name\": \"project 1\", \"authors\": \"Authors\", \"created\": \"2023-
↪04-12 21:23:28.625859\", \"owner_id\": 15}]"
```

CHAPTER

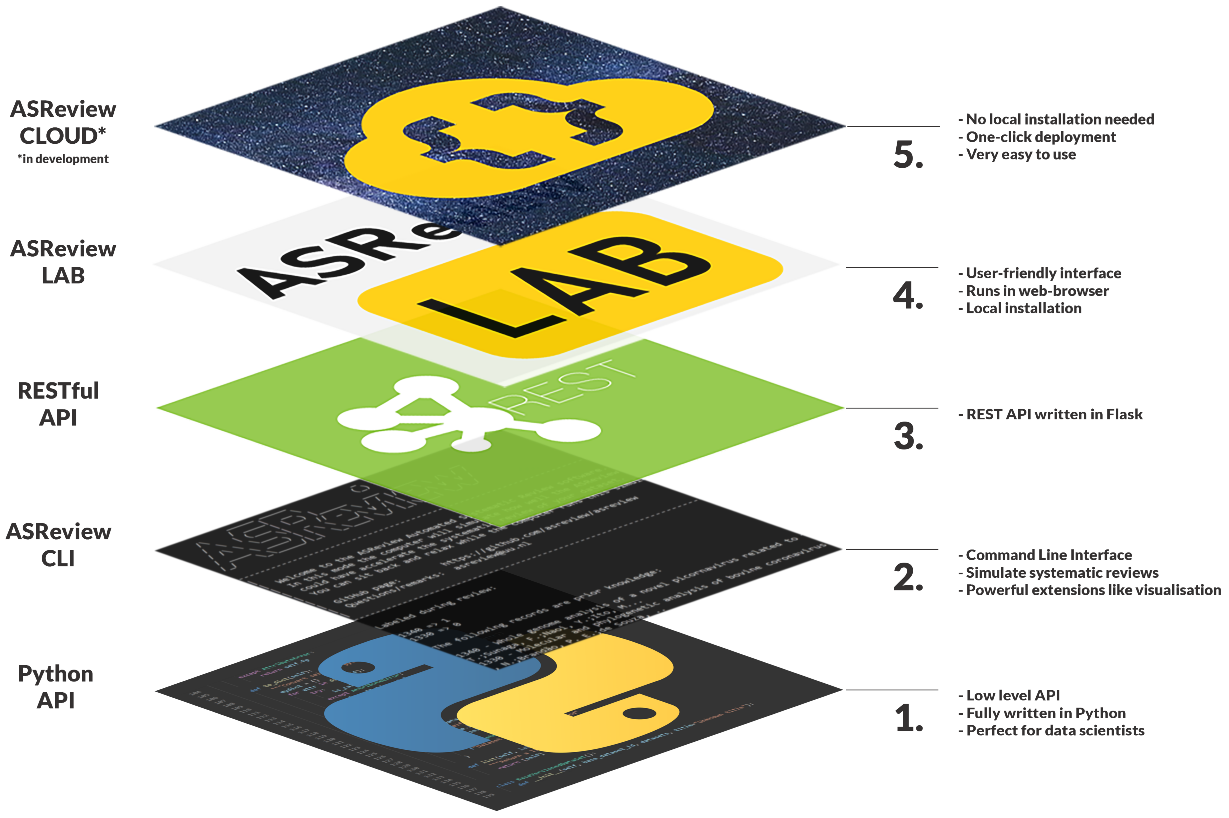
# TWENTYTWO

# OVERVIEW

The development section is meant for users that need advanced functions of ASReview LAB and for developers. It contains technical information on the usage, instructions for developing extensions, and an extensive API reference.

## 22.1 ASReview architecture

ASReview provides users an API to interact directly with the underlying ASReview machinery. This provides researchers an interface to study the behavior of algorithms and develop custom workflows. The following overview shows the available interfaces for interacting with the ASReview software:


ASReview CLOUD*
*in development
5.
- No local installation needed
- One-click deployment
- Very easy to use
ASReview LAB
4.
- User-friendly interface
- Runs in web-browser
- Local installation
RESTful API
3.
- REST API written in Flask
ASReview CLI
2.
- Command Line Interface
- Simulate systematic reviews
- Powerful extensions like visualisation
Python API
1.
- Low level API
- Fully written in Python
- Perfect for data scientists

- Layer 5: ASReview CLOUD

  - ASReview is currently in development. For information on ASReview CLOUD, be sure visit our communication channels.

- Layer 4: *ASReview LAB*
  - ASReview LAB is the user friendly webapp and all underlying interfaces. Documentation on LAB can be found in the *ASReview LAB section*.
- Layer 3: REST API
  - The REST API uses a Flask REST API to provide a method to let the React webapp communicate with the backend and algorithms. The REST API is not documented and should be considered 'internal use only'.
- Layer 2: *Command Line*
  - The Command Line is an interface used to open ASReview LAB, run simulations, and run *Subcommand extensions* for ASReview. This development section documents all available command line options for both *ASReview LAB* and *simulation mode*.
- Layer 1: *API Reference*
  - The ASReview API is a low level Python interface for ASReview. This interface requires detailed knowledge about the workings of the software. This reference contains extensive documentation on all functions, classes, and modules found in ASReview.
  - An outline for usage can be found in *Simulate with Python API* and *Access data from ASReview file*.

## 22.2 Extensions

*The Create an extension* section documents the creation of model, subcommand, and dataset extensions for ASReview. More information on extensions can be found in the extension *Extensions*.

CHAPTER

# TWENTYTHREE

# COMMAND LINE

ASReview provides a powerful command line interface for running tasks *Start ASReview LAB* and *Simulation via command line*. Also *Extensions* often make use of the command line interface by extending it with subcommands.

The structure of the command line is given by:

```
asreview [-h] [-V] [subcommand]
```

A list of available and installed subcommands is given by `asreview -h`. Each subcommand is listed with its subcommand, the package it comes from and the version of the package. For example, the default subcommand `lab` (to start ASReview LAB) is given by listed as `lab [asreview-1.0]`. An subcommand installed via an extension, e.g. `plot`, is listed as `plot [asreview-insights-1.1]` where `asreview-insights` is the name of the extension that installed this subcommand and 1.1 is the version of this package.

CHAPTER

# TWENTYFOUR

# ACCESS DATA FROM ASREVIEW FILE

The API is still under development and can change at any time without warning.

Data generated using ASReview LAB is stored in an ASReview project file. Via the ASReview Python API, there are two ways to access the data in the ASReview (extension `.asreview`) file: Via the *project-API* and the *state-API*. The project API is for retrieving general project settings, the imported dataset, the feature matrix, etc. The state API retrieves data related directly to the reviewing process, such as the labels, the time of labeling, and the classifier used.

## 24.1 Example Data

To illustrate the ASReview Python API, the benchmark dataset `van_de_Schoot_2017` is used. The project file `example.asreview` can be obtained by running `asreview simulate benchmark:van_de_Schoot_2017 -s example.asreview --seed 101`.

The ASReview Python API can be used for project files obtained via the Oracle, Validation, and Simulation mode.

## 24.2 Python Imports

[1]:
```python
import shutil
from pathlib import Path

import pandas as pd
from asreview import open_state
from asreview import ASReviewProject
from asreview import ASReviewData
```

## 24.3 Project API

The ASReview project file is a zipped folder. To unzip the folder and store its contents in a temporary directory, use the following code:

```
[2]: project_path = Path("tmp_data")
     project_path.mkdir()
     project = ASReviewProject.load("example.asreview", project_path)
```

The returned `project` instance is of type *ASReviewProject*.

To inspect the project details, use the following code:

```
[3]: project.config
```

```
[3]: {'version': '1.2+6.g41c4257.dirty',
      'id': 'example',
      'mode': 'simulate',
      'name': 'example',
      'description': 'Simulation created via ASReview via command line interface',
      'authors': None,
      'created_at_unix': 1683798551,
      'datetimeCreated': '2023-05-11 11:49:11.327073',
      'reviews': [{'id': 'e611d2cbd89b401aa376fa4eca1c517e',
        'start_time': '2023-05-11 11:49:12.323797',
        'status': 'finished',
        'end_time': '2023-05-11 11:49:32.450593'}],
      'feature_matrices': [{'id': 'tfidf', 'filename': 'tfidf_feature_matrix.npz'}],
      'dataset_path': 'van_de_Schoot_2017.csv'}
```

The imported dataset is located at `/tmp_data/{project_name}/data/{dataset_filename}`, and can be inspected using the following code:

```
[4]: dataset_fp = Path(
         project_path, project.config["id"], "data", project.config["dataset_path"]
     )
     dataset = ASReviewData.from_file(dataset_fp)
     print(f"The dataset contains {len(dataset)} records.")
     dataset.to_dataframe().head()
```

```
     The dataset contains 6189 records.
```

```
[4]:                                                            title
     record_id
     0                Manual for ASEBA School-Age Forms & Profiles  \
     1           Queensland Trauma Registry: A summary of paedi...
     2           Posttraumatic Stress Disorder: Scientific and ...
     3                                  SOCIAL CLASS AND MENTAL ILLNESS
     4           Computerised test generation for cross-nationa...

                                                            abstract keywords
     record_id
```

(continues on next page)

(continued from previous page)

```
0                                                                    \
1
2          This comprehensive overview of research and cl...
3
4          “‘Computerised Test Generation for Cross-Natio...

                                                   authors    year  date
record_id
0                         Achenbach, T. M., Rescorla, L. A.  2001.0  2001  \
1                         Dallow, N., Lang, J., Bellamy, N.  2007.0  2007
2          Ford, J. D., Grasso, D. J., Elhai, J. D., Cour...  2015.0   NaN
3                       Hollingshead, A. B., Redlich, F. C.  1958.0   NaN
4                                            Irvine, S. H.  2014.0   NaN

           doi label_included  label_abstract_screening  duplicate_record_id
record_id
0          NaN              0                         0                  NaN
1          NaN              0                         0                  NaN
2          NaN              0                         0                  NaN
3          NaN              0                         0                  NaN
4          NaN              0                         0                  NaN
```

To obtain the content of the feature matrix, for example, the first row of the matrix, use the following code (note the matrix is in a sparse matrix format):

[5]:
```
feature_extraction_id = project.feature_matrices[0]["id"]
feature_matrix = project.get_feature_matrix(feature_extraction_id)
print(feature_matrix[0])
```

```
  (0, 20452)    0.35937211648312967
  (0, 18297)    0.26158369118434677
  (0, 13842)    0.3248271421716685
  (0, 9739)     0.38355660008860293
  (0, 3231)     0.7059309068495663
  (0, 2384)     0.22684547910949254
```

## 24.4 State API

The data stored during the review process can be accessed as a pandas DataFrame using the following code:

[6]:
```
with open_state("example.asreview") as state:
    df = state.get_dataset()
    print(f"The state contains {len(df)} records.")
```

```
The state contains 561 records.
```

The returned `state` instance is of type *SQLiteState*. Note that the state contains less records than the original dataset. This is because by default the simulation stops after finding all relevant records.

```
[7]: df.to_csv(project_path / "example_state.csv", index=False)
     df.head()
```

```
[7]:    record_id  label classifier query_strategy balance_strategy  \
     0       4435      1       None          prior             None
     1       5560      0       None          prior             None
     2       4434      1         nb            max           double
     3       3668      0         nb            max           double
     4       3142      0         nb            max           double

       feature_extraction  training_set               labeling_time notes
     0               None            -1  2023-05-11 11:49:16.186034  None
     1               None            -1  2023-05-11 11:49:16.186034  None
     2              tfidf             2  2023-05-11 11:49:16.420695  None
     3              tfidf             3  2023-05-11 11:49:16.444989  None
     4              tfidf             4  2023-05-11 11:49:16.505603  None
```

You can merge the information from the state file with the original dataset.

```
[8]: df["labeling_order"] = df.index
     dataset_with_results = dataset.df.join(df.set_index("record_id"))
     dataset_with_results.to_csv(project_path / "data_and_state_merged.csv", index=False)
     dataset_with_results
```

```
[8]:                                                        title  \
     record_id
     0                Manual for ASEBA School-Age Forms & Profiles
     1           Queensland Trauma Registry: A summary of paedi...
     2           Posttraumatic Stress Disorder: Scientific and ...
     3                                SOCIAL CLASS AND MENTAL ILLNESS
     4           Computerised test generation for cross-nationa...
     ...                                                      ...
     6184        Biological and clinical framework for posttrau...
     6185        Dividing traffic sub-areas based on a parallel...
     6186        Quantifying resilience to enhance individualiz...
     6187        A discriminant analysis of variables related t...
     6188        Developmental trajectories of pain/disability ...

                                                         abstract  \
     record_id
     0
     1
     2           This comprehensive overview of research and cl...
     3
     4           “‘Computerised Test Generation for Cross-Natio...
     ...                                                      ...
     6184        Three decades of posttraumatic stress disorder...
     6185        In order to alleviate the traffic congestion a...
     6186        Resilience is the human ability to adapt in th...
     6187
     6188
```

(continues on next page)

(continued from previous page)

```
                                                   keywords
record_id
0                                                             \
1
2
3
4
...                                                     ...
6184
6185       GPS trajectories, K-means, MapReduce, Traffic ...
6186       Adaptation, Autonomic Nervous System, Resilien...
6187
6188

                                                    authors    year  date
record_id
0                           Achenbach, T. M., Rescorla, L. A.  2001.0  2001  \
1                           Dallow, N., Lang, J., Bellamy, N.  2007.0  2007
2          Ford, J. D., Grasso, D. J., Elhai, J. D., Cour...  2015.0   NaN
3                         Hollingshead, A. B., Redlich, F. C.  1958.0   NaN
4                                              Irvine, S. H.  2014.0   NaN
...                                                     ...     ...   ...
6184                             Vermetten, E., Lanius, R. A.  2012.0   NaN
6185       Wang, B., Tao, L., Gao, C., Xia, D., Rong, Z.,...  2014.0   NaN
6186       Winslow, B., Carroll, M., Jones, D., Hannigan,...  2013.0   NaN
6187                                          Frye, James S.  1981.0   NaN
6188                Sterling, M., Hendrikz, J., Kenardy, J.  2010.0   NaN

           doi  label_included  label_abstract_screening  duplicate_record_id
record_id
0          NaN               0                         0                  NaN  \
1          NaN               0                         0                  NaN
2          NaN               0                         0                  NaN
3          NaN               0                         0                  NaN
4          NaN               0                         0                  NaN
...        ...             ...                       ...                  ...
6184       NaN               0                         0                  NaN
6185       NaN               0                         0                  NaN
6186       NaN               0                         0                  NaN
6187       NaN               0                         1                  NaN
6188       NaN               0                         1                  NaN

           label classifier query_strategy balance_strategy
record_id
0            NaN        NaN            NaN              NaN  \
1            NaN        NaN            NaN              NaN
2            NaN        NaN            NaN              NaN
3            NaN        NaN            NaN              NaN
4            NaN        NaN            NaN              NaN
...          ...        ...            ...              ...
6184         NaN        NaN            NaN              NaN
6185         NaN        NaN            NaN              NaN
```

(continues on next page)

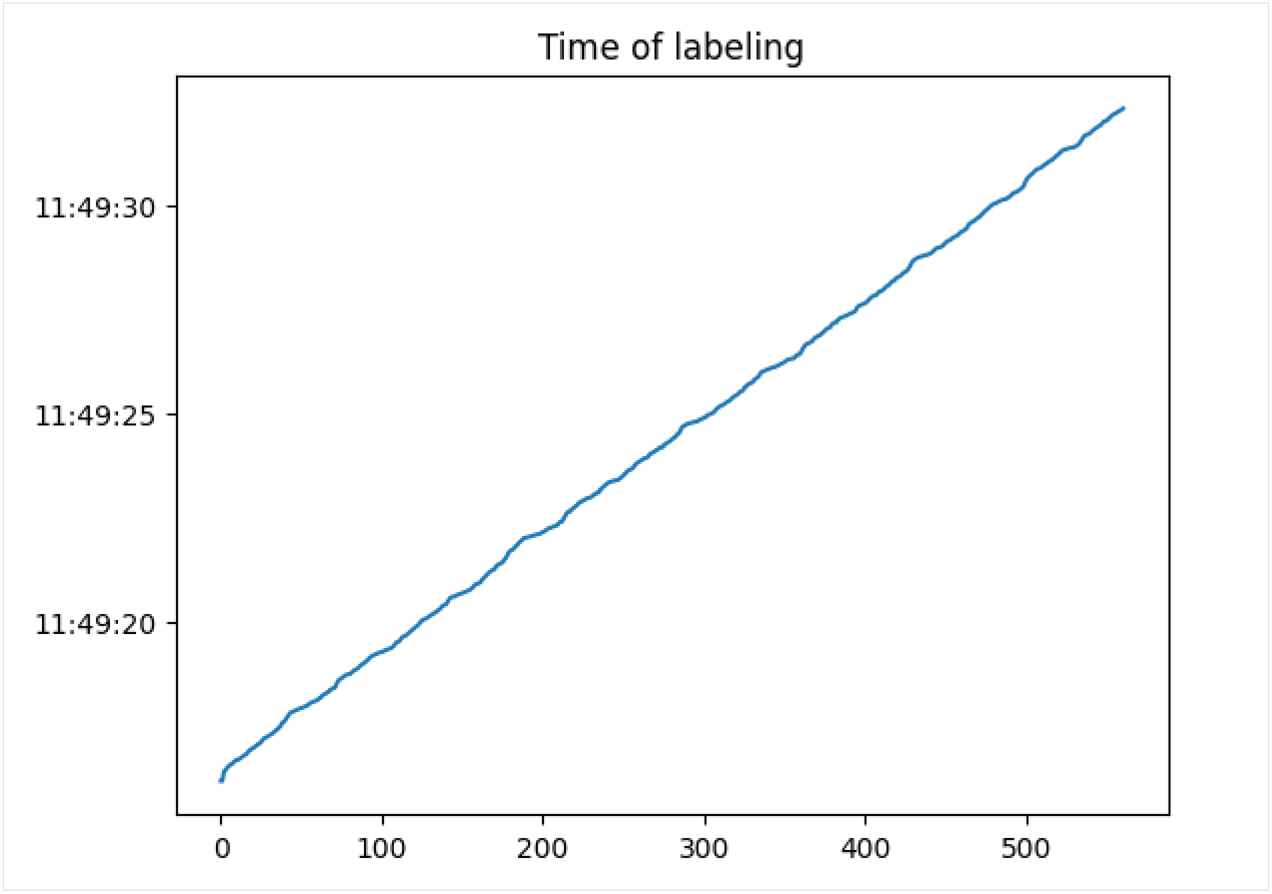
(continued from previous page)

```
6186         0.0         nb            max           double
6187         NaN        NaN            NaN              NaN
6188         0.0         nb            max           double

          feature_extraction  training_set              labeling_time notes  
record_id                                                                    
0                        NaN           NaN                        NaN   NaN  \
1                        NaN           NaN                        NaN   NaN  
2                        NaN           NaN                        NaN   NaN  
3                        NaN           NaN                        NaN   NaN  
4                        NaN           NaN                        NaN   NaN  
...                      ...           ...                        ...   ...  
6184                     NaN           NaN                        NaN   NaN  
6185                     NaN           NaN                        NaN   NaN  
6186                   tfidf         535.0  2023-05-11 11:49:31.593247  None  
6187                     NaN           NaN                        NaN   NaN  
6188                   tfidf         333.0  2023-05-11 11:49:25.857883  None  

           labeling_order  
record_id                  
0                     NaN  
1                     NaN  
2                     NaN  
3                     NaN  
4                     NaN  
...                   ...  
6184                  NaN  
6185                  NaN  
6186                535.0  
6187                  NaN  
6188                333.0  

[6189 rows x 19 columns]
```

There are also multiple functions to obtain one specific variable in the data. For example, to plot the labeling times in a graph, use the following code:

[9]:
```
with open_state("example.asreview") as state:
    labeling_times = state.get_labeling_times()
pd.to_datetime(labeling_times).plot(title="Time of labeling")
```

[9]:
```
<Axes: title={'center': 'Time of labeling'}>
```



By default, the records that are part of the prior knowledge are included in the results. To obtain the labels use the following code:

[10]:
```
with open_state("example.asreview") as state:
    labels = state.get_labels(priors=False)
labels
```

[10]:
```
0      1
1      0
2      0
3      0
4      0
      ..
554    0
555    0
556    0
557    0
558    1
Name: label, Length: 559, dtype: int64
```

To obtain the data corresponding to a specific record identifier, use the following code:

[11]:
```
with open_state("example.asreview") as state:
    record_data = state.get_data_by_record_id(5176)
```

(continues on next page)

(continued from previous page)

```
record_data
```

[11]:
```
   record_id  label classifier query_strategy balance_strategy  
0       5176      0         nb            max           double  \

  feature_extraction  training_set               labeling_time notes
0              tfidf            29  2023-05-11 11:49:17.247842  None
```

To obtain all settings used for the project, run the following code:

[12]:
```
with open_state("example.asreview") as state:
    settings = state.settings_metadata
settings
```

[12]:
```
{'settings': {'model': 'nb',
  'query_strategy': 'max',
  'balance_strategy': 'double',
  'feature_extraction': 'tfidf',
  'n_instances': 1,
  'stop_if': 'min',
  'n_prior_included': 1,
  'n_prior_excluded': 1,
  'model_param': {'alpha': 3.822},
  'query_param': {},
  'feature_param': {'ngram_max': 1,
   'stop_words': 'english',
   'split_ta': 0,
   'use_keywords': 0},
  'balance_param': {'a': 2.155, 'alpha': 0.94, 'b': 0.789, 'beta': 1.0}},
 'state_version': '1',
 'software_version': '1.2+6.g41c4257.dirty',
 'model_has_trained': True}
```

The state also contains the ranking and the relevance score (if the model uses relevance scores) of the last iteration of the machine learning model. To get these, use the following code:

[13]:
```
with open_state("example.asreview") as state:
    last_ranking = state.get_last_ranking()
    last_probabilities = state.get_last_probabilities()
print("RANKING:")
print(last_ranking[["record_id", "ranking"]])
print("RELEVANCE SCORES:")
print(last_probabilities)
```

```
RANKING:
      record_id  ranking
0          2445        0
1          2446        1
2          2444        2
3           720        3
4           719        4
```

(continues on next page)

(continued from previous page)

```
...         ...      ...
6184        1766     6184
6185          63     6185
6186        4427     6186
6187        2851     6187
6188        4888     6188

[6189 rows x 2 columns]
RELEVANCE SCORES:
0       0.637417
1       0.671088
2       0.707728
3       0.777025
4       0.672183
          ...
6184    0.792209
6185    0.697030
6186    0.828880
6187    0.768638
6188    0.844882
Name: proba, Length: 6189, dtype: float64
```

## 24.5 Cleanup

The following code removes the temporary folder that was created:

[14]:
```
shutil.rmtree(project_path)
```

CHAPTER

# TWENTYFIVE

# EXTENSIONS

ASReview extensions enable you to integrate your programs with the ASReview framework seamlessly, by using the Python API. These extensions fall into three different categories, and interact with the API in different ways.

1. *Model extensions*
2. *Subcommand extensions*
3. *Dataset extensions*

The extensibility of the framework is provided by the entrypoints of setuptools. You will need to create a package and install it (for example with pip).

Did you develop a useful extension to ASReview and want to list it on the Discussion platform? Create a Pull Request or open an issue on GitHub.

For more information on the ASReview API for creating an extension, a technical reference for development is found under the *API reference*. This technical reference contains functions for use in your extension, and an overview of all classes to extend on.

## 25.1 Model Extensions

An extension of a `asreview.models.base.BaseModel` type class.

Model extensions extent the ASReview software with new classifiers, query strategies, balance strategies, or feature extraction techniques. These extensions extend one of the model base classes (`asreview.models.balance.base`, `asreview.models.classifiers.base`, `asreview.models.feature_extraction.base`, `asreview.models.query.base`).

The easiest way to extend ASReview with a model is by using the . Create a copy of the template and add the new algorithm to a new model file. It is advised to use the following structure of the package:

```
├── README.md
├── asreviewcontrib
│   └── models
│       ├── classifiers
│       │   ├── __init__.py
│       │   └── example_model.py
│       ├── feature_extraction
│       │   ├── __init__.py
│       │   └── example_feature_extraction.py
│       ├── balance
│       │   ├── __init__.py
│       │   └── example_balance_strategies.py
│       │
```

(continues on next page)

(continued from previous page)

```
│       └── query
│           ├── __init__.py
│           └── example_query_strategies.py
├── setup.py
└── tests
```

The next step is to add metadata to the setup.py file. Edit the `name` of the package and point the `entry_points` to the models.

```
entry_points={
    'asreview.models.classifiers': [
        'example = asreviewcontrib.models.classifiers.example_model:ExampleClassifier',
    ],
    'asreview.models.feature_extraction': [
        # define feature_extraction algorithms
    ],
    'asreview.models.balance': [
        # define balance_strategy algorithms
    ],
    'asreview.models.query': [
        # define query_strategy algorithms
    ]
},
```

This code registers the model with name `example`.

## 25.2 Subcommand Extensions

An extension of the `asreview.entry_points.base.BaseEntryPoint` class.

Subcommand extensions are programs that create a new entry point for ASReview. From this entry point the Python API can be used in many ways (like `plot` or `simulate`).

Extensions in ASReview are Python packages and can extend the subcommands of asreview (see `asreview -h`). An example of a subcommand extension is ASReview Insights.

The easiest way to create a new subcommand is by defining a class that can be used as a new entry point for ASReview. This class should inherit from `asreview.entry_points.base.BaseEntryPoint`. Add the functionality to the class method `execute`.

```
from asreview.entry_points import BaseEntryPoint

class ExampleEntryPoint(BaseEntryPoint):

    description = "Description of example extension"
    extension_name = "asreview-example"  # Name of the extension
    version = "1.0"  # Version of the extension in x.y(.z) format.

    def execute(self, argv):
        pass  # Implement your functionality here.
```

It is strongly recommended to define the attributes `description`, `extension_name`, and `version`.

The class method `execute` accepts a positional argument (`argv` in this example). First create the functionality you would like to be able to use in any directory. The argument `argv` are the command line arguments left after removing asreview and the entry point.

It is advised to place the newly defined class `ExampleEntryPoints` in the following package structure: `asreviewcontrib.{extension_name}.{your_modules}`. For example:

```
├── README.md
├── asreviewcontrib
│   └── example
│       ├── __init__.py
│       ├── entrypoint.py
│       └── example_utils.py
├── setup.py
└── tests
```

Create a `setup.py` in the root of the package, and set the keyword argument *entry_points* of `setup()` under `asreview.entry_points`, for example:

```
entry_points={
    "asreview.entry_points": [
        "example = asreviewcontrib.example.entrypoint:ExampleEntryPoint",
    ]
}
```

After installing this package, ASReview is extended with the `asreview example` subcommand. See `asreview -h` for this option.

## 25.3 Dataset Extensions

An extension of the *`asreview.datasets.BaseDataSet`* class.

Dataset extensions integrate new datasets for use in ASReview. Adding datasets via extension provides quick access to the dataset via Command Line Interface or in ASReview LAB.

It is advised to place the new dataset `your_dataset` in the following package structure:

```
├── README.md
├── asreviewcontrib
│   └── dataset_name
│       ├── __init__.py
│       └── your_dataset.py
├── data
│   └── your_dataset.csv
├── setup.py
└── tests
```

For minimal functionality, `your_dataset.py` should extent *`asreview.datasets.BaseDataSet`* and *`asreview.datasets.BaseDataGroup`*.

A working template to clone and use can be found at Template for extending ASReview with a new dataset.

Further functionality can be extensions of any other class in *`asreview.datasets`*.

CHAPTER

# TWENTYSIX

# API REFERENCE

## 26.1 Data and datasets

### 26.1.1 Dataset object

| | |
|---|---|
| *load_dataset*(name, **kwargs) | Load data from file, URL, or plugin. |
| *Dataset*([df, column_spec]) | Dataset object to the dataset with texts, labels, DOIs etc. |

#### asreview.load_dataset

`asreview.load_dataset`(*name*, ***kwargs*)

Load data from file, URL, or plugin.

**Parameters**

- **name** (*str*, *pathlib.Path*) – File path, URL, or alias of extension dataset.
- ****kwargs** – Keyword arguments passed to the reader.

**Returns**

*asreview.Dataset* – Inititalized ASReview data object.

#### asreview.Dataset

`class asreview.Dataset`(*df=None*, *column_spec=None*)

Dataset object to the dataset with texts, labels, DOIs etc.

**Parameters**

- **df** (*pandas.DataFrame*) – Dataframe containing the data for the ASReview data object.
- **column_spec** (*dict*) – Specification for which column corresponds to which standard specification. Key is the standard specification, key is which column it is actually in. Default: None.

**Variables**

- **record_ids** (*numpy.ndarray*) – Return an array representing the data in the Index.
- **texts** (*numpy.ndarray*) – Returns an array with either headings, bodies, or both.
- **headings** (*numpy.ndarray*) – Returns an array with dataset headings.

- **title** (*numpy.ndarray*) – Identical to headings.
- **bodies** (*numpy.ndarray*) – Returns an array with dataset bodies.
- **abstract** (*numpy.ndarray*) – Identical to bodies.
- **notes** (*numpy.ndarray*) – Returns an array with dataset notes.
- **keywords** (*numpy.ndarray*) – Returns an array with dataset keywords.
- **authors** (*numpy.ndarray*) – Returns an array with dataset authors.
- **doi** (*numpy.ndarray*) – Returns an array with dataset DOI.
- **included** (*numpy.ndarray*) – Returns an array with document inclusion markers.
- **final_included** (*numpy.ndarray*) – Pending deprecation! Returns an array with document inclusion markers.
- **labels** (*numpy.ndarray*) – Identical to included.

## Attributes

| |
|---|
| *abstract* |
| *authors* |
| *bodies* |
| *doi* |
| *headings* |
| *included* |
| *keywords* |
| *labels* |
| *notes* |
| *record_ids* |
| *texts* |
| *title* |
| *url* |

### asreview.Dataset.abstract

`property Dataset.abstract`

### asreview.Dataset.authors

`property Dataset.authors`

### asreview.Dataset.bodies

`property Dataset.bodies`

### asreview.Dataset.doi

`property Dataset.doi`

### asreview.Dataset.headings

`property Dataset.headings`

### asreview.Dataset.included

`property Dataset.included`

### asreview.Dataset.keywords

`property Dataset.keywords`

### asreview.Dataset.labels

`property Dataset.labels`

### asreview.Dataset.notes

`property Dataset.notes`

### asreview.Dataset.record_ids

**property** Dataset.**record_ids**

### asreview.Dataset.texts

**property** Dataset.**texts**

### asreview.Dataset.title

**property** Dataset.**title**

### asreview.Dataset.url

**property** Dataset.**url**

## Methods

| | |
|---|---|
| *drop_duplicates*([pid, inplace, reset_index]) | Drop duplicate records. |
| *duplicated*([pid]) | Return boolean Series denoting duplicate rows. |
| *get*(name) | Get column with name. |
| *is_prior*() | Get the labels that are marked as 'prior'. |
| *record*(i) | Create a record from an index. |
| *to_dataframe*([labels, ranking, keep_old_labels]) | Create new dataframe with updated label (order). |
| *to_file*(fp[, labels, ranking, writer, ...]) | Export data object to file. |

### asreview.Dataset.drop_duplicates

Dataset.**drop_duplicates**(*pid='doi'*, *inplace=False*, *reset_index=True*)

Drop duplicate records.

Drop duplicates based on titles and abstracts and if available, on a persistent identifier (PID) such the Digital Object Identifier (DOI).

**Parameters**

- **pid** (*string, default 'doi'*) – Which persistent identifier to use for deduplication.
- **inplace** (*boolean, default False*) – Whether to modify the DataFrame rather than creating a new one.
- **reset_index** (*boolean, default True*) – If True, the existing index column is reset to the default integer index.

**Returns**

*pandas.DataFrame or None* – DataFrame with duplicates removed or None if inplace=True

### asreview.Dataset.duplicated

`Dataset.duplicated(pid='doi')`

Return boolean Series denoting duplicate rows.

Identify duplicates based on titles and abstracts and if available, on a persistent identifier (PID) such as the Digital Object Identifier (DOI).

**Parameters**
**pid** (*string*) – Which persistent identifier to use for deduplication. Default is 'doi'.

**Returns**
*pandas.Series* – Boolean series for each duplicated rows.

### asreview.Dataset.get

`Dataset.get(name)`

Get column with name.

### asreview.Dataset.is_prior

`Dataset.is_prior()`

Get the labels that are marked as 'prior'.

**Returns**
*numpy.ndarray* – Array of booleans that have the 'prior' property.

### asreview.Dataset.record

`Dataset.record(i)`

Create a record from an index.

**Parameters**
**i** (*int, iterable*) – Index of the record, or list of indices.

**Returns**
*Record* – The corresponding record if i was an integer, or a list of records if i was an iterable.

### asreview.Dataset.to_dataframe

`Dataset.to_dataframe(labels=None, ranking=None, keep_old_labels=False)`

Create new dataframe with updated label (order).

**Parameters**

- **labels** (*list, numpy.ndarray*) – Current labels will be overwritten by these labels (including unlabelled). No effect if labels is None.
- **ranking** (*list*) – Reorder the dataframe according to these record_ids. Default ordering if ranking is None.
- **keep_old_labels** (*bool*) – If True, the old labels are kept in a column 'asreview_label_to_validate'. Default False.

**Returns**
*pandas.DataFrame* – Dataframe of all available record data.

### asreview.Dataset.to_file

`Dataset.to_file(fp, labels=None, ranking=None, writer=None, keep_old_labels=False)`

Export data object to file.

RIS, CSV, TSV and Excel are supported file formats at the moment.

**Parameters**

- **fp** (*str*) – Filepath to export to.
- **labels** (*list, numpy.ndarray*) – Labels to be inserted into the dataframe before export.
- **ranking** (*list, numpy.ndarray*) – Optionally, dataframe rows can be reordered.
- **writer** (*class*) – Writer to export the file.
- **keep_old_labels** (*bool*) – If True, the old labels are kept in a column 'asreview_label_to_validate'. Default False.

## 26.1.2 Readers and writers

Functions and classes for file reading and writing.

| | |
|---|---|
| *data.list_readers*() | List available dataset reader classes. |
| *data.list_writers*() | List available dataset writer classes. |
| *data.CSVReader*() | CVS file reader. |
| *data.CSVWriter*() | CSV file writer. |
| *data.ExcelReader*() | Excel file reader. |
| *data.ExcelWriter*() | Excel file writer. |
| *data.RISReader*() | RIS file reader. |
| *data.RISWriter*() | RIS file writer. |
| *data.TSVWriter*() | TSV file writer. |

### asreview.data.list_readers

`asreview.data.list_readers()`

List available dataset reader classes.

**Returns**
*list* – Classes of available dataset readers in alphabetical order.

### asreview.data.list_writers

`asreview.data.list_writers()`

List available dataset writer classes.

**Returns**
*list* – Classes of available dataset writers in alphabetical order.

### asreview.data.CSVReader

`class asreview.data.CSVReader`

CVS file reader.

#### Attributes

| |
|---|
| *read_format* |
| *write_format* |

##### asreview.data.CSVReader.read_format

`CSVReader.read_format = ['.csv', '.tab', '.tsv']`

##### asreview.data.CSVReader.write_format

`CSVReader.write_format = ['.csv', '.tsv', '.xlsx']`

#### Methods

| | |
|---|---|
| *read_data*(fp) | Import dataset. |

##### asreview.data.CSVReader.read_data

`classmethod CSVReader.read_data(fp)`

Import dataset.

**Parameters**
**fp** (*str*, *pathlib.Path*) – File path to the CSV file.

**Returns**
*list* – List with entries.

## asreview.data.CSVWriter

`class asreview.data.CSVWriter`

CSV file writer.

### Attributes

| |
|---|
| *label* |
| *name* |
| *write_format* |

#### asreview.data.CSVWriter.label

`CSVWriter.label = 'CSV (UTF-8)'`

#### asreview.data.CSVWriter.name

`CSVWriter.name = 'csv'`

#### asreview.data.CSVWriter.write_format

`CSVWriter.write_format = '.csv'`

### Methods

| | |
|---|---|
| *write_data*(df, fp[, sep]) | Export dataset. |

#### asreview.data.CSVWriter.write_data

`classmethod CSVWriter.write_data(df, fp, sep=',')`

Export dataset.

**Parameters**

- **df** (*pandas.Dataframe*) – Dataframe of all available record data.
- **fp** (*str, NoneType*) – Filepath or None for buffer.
- **sep** (*str*) – Seperator of the file.

**Returns**

*CSV file* – Dataframe of all available record data.

### asreview.data.ExcelReader

`class asreview.data.ExcelReader`

Excel file reader.

#### Attributes

| | |
|---|---|
| *read_format* | |
| *write_format* | |

##### asreview.data.ExcelReader.read_format

`ExcelReader.read_format = ['.xlsx']`

##### asreview.data.ExcelReader.write_format

`ExcelReader.write_format = ['.csv', '.tsv', '.xlsx']`

#### Methods

| | |
|---|---|
| *read_data*(fp) | Import dataset. |

##### asreview.data.ExcelReader.read_data

`classmethod ExcelReader.read_data(fp)`

Import dataset.

**Parameters**
**fp** (*str*, *pathlib.Path*) – File path to the Excel file (.xlsx).

**Returns**
*list* – List with entries.

### asreview.data.ExcelWriter

`class asreview.data.ExcelWriter`

Excel file writer.

### Attributes

| | |
|---|---|
| *label* | |
| *name* | |
| *write_format* | |

#### asreview.data.ExcelWriter.label

`ExcelWriter.label = 'Excel'`

#### asreview.data.ExcelWriter.name

`ExcelWriter.name = 'xlsx'`

#### asreview.data.ExcelWriter.write_format

`ExcelWriter.write_format = '.xlsx'`

### Methods

| | |
|---|---|
| *write_data*(df, fp) | Export dataset. |

#### asreview.data.ExcelWriter.write_data

**classmethod** `ExcelWriter.`**`write_data`**`(`*df*, *fp*`)`

Export dataset.

**Parameters**

- **df** (*pandas.Dataframe*) – Dataframe of all available record data.
- **fp** (*str, NoneType*) – Filepath or None for buffer.

**Returns**

*Excel file* – Dataframe of all available record data.

### asreview.data.RISReader

`class asreview.data.RISReader`

RIS file reader.

#### Attributes

| | |
|---|---|
| *read_format* | |
| *write_format* | |

##### asreview.data.RISReader.read_format

`RISReader.read_format = ['.ris', '.txt']`

##### asreview.data.RISReader.write_format

`RISReader.write_format = ['.csv', '.tsv', '.xlsx', '.ris']`

#### Methods

| | |
|---|---|
| *read_data*(fp) | Import dataset. |

##### asreview.data.RISReader.read_data

`classmethod RISReader.read_data(fp)`

Import dataset.

**Parameters**

- **fp** (*str*, *pathlib.Path*) – File path to the RIS file.
- **note_list** (*list*) – A list of notes, coming from the Dataframe's "notes" column.

**Returns**

*pd.DataFrame* – Dataframe with entries.

**Raises**

`ValueError` – File with unrecognized encoding is used as input.

## asreview.data.RISWriter

`class asreview.data.RISWriter`

RIS file writer.

### Attributes

| |
|---|
| *caution* |
| *label* |
| *name* |
| *write_format* |

#### asreview.data.RISWriter.caution

`RISWriter.caution = 'Available only if you imported a RIS file when creating the project'`

#### asreview.data.RISWriter.label

`RISWriter.label = 'RIS'`

#### asreview.data.RISWriter.name

`RISWriter.name = 'ris'`

#### asreview.data.RISWriter.write_format

`RISWriter.write_format = '.ris'`

### Methods

| | |
|---|---|
| *write_data*(df, fp) | Export dataset. |

#### asreview.data.RISWriter.write_data

**classmethod** RISWriter.**write_data**(*df*, *fp*)

Export dataset.

**Parameters**

- **df** (*pd.Dataframe*) – Dataframe of all available record data.
- **fp** (*str, pathlib.Path*) – File path to the RIS file, if exists.

**Returns**

*RIS file* – Dataframe of all available record data.

### asreview.data.TSVWriter

**class** asreview.data.**TSVWriter**

TSV file writer.

#### Attributes

| |
|---|
| *label* |
| *name* |
| *write_format* |

#### asreview.data.TSVWriter.label

`TSVWriter.label = 'TSV (UTF-8)'`

#### asreview.data.TSVWriter.name

`TSVWriter.name = 'tsv'`

#### asreview.data.TSVWriter.write_format

`TSVWriter.write_format = '.tsv'`

#### Methods

| | |
|---|---|
| *write_data*(df, fp[, sep]) | Export dataset. |

#### asreview.data.TSVWriter.write_data

**classmethod** `TSVWriter.`**`write_data`**(*df*, *fp*, *sep='\t'*)

Export dataset.

**Parameters**

- **df** (*pandas.Dataframe*) – Dataframe of all available record data.
- **fp** (*str, NoneType*) – Filepath or None for buffer.
- **sep** (*str*) – Seperator of the file.

**Returns**

*TSV file* – Dataframe of all available record data.

### 26.1.3 Statistics

| | |
|---|---|
| *data.statistics.abstract_length*(data) | Return the average length of the abstracts. |
| *data.statistics.n_duplicates*(data[, pid]) | Number of duplicates. |
| *data.statistics.n_irrelevant*(data) | Return the number of irrelevant records. |
| *data.statistics.n_keywords*(data) | Return the number of keywords. |
| *data.statistics.n_missing_abstract*(data) | Return the number of records with missing abstracts. |
| *data.statistics.n_missing_title*(data) | Return the number of records with missing titles. |
| *data.statistics.n_records*(data) | Return the number of records. |
| *data.statistics.n_relevant*(data) | Return the number of relevant records. |
| *data.statistics.n_unlabeled*(data) | Return the number of unlabeled records. |
| *data.statistics.title_length*(data) | Return the average length of the titles. |

#### asreview.data.statistics.abstract_length

`asreview.data.statistics.`**`abstract_length`**(*data*)

Return the average length of the abstracts.

**Parameters**

**data** (`asreview.Dataset`) – An Dataset object with the records.

**Returns**

*int* – The statistic

### asreview.data.statistics.n_duplicates

`asreview.data.statistics.`**`n_duplicates`**`(`*data*, *pid='doi'*`)`

Number of duplicates.

Duplicate detection can be a very challenging task. Multiple algorithms can be used and results can be vary.

**Parameters**

- **`data`** (`asreview.Dataset`) – An Dataset object with the records.
- **`pid`** (*`string`*) – Which persistent identifier (PID) to use for deduplication. Default is 'doi'.

**Returns**

*int* – Number of duplicates

### asreview.data.statistics.n_irrelevant

`asreview.data.statistics.`**`n_irrelevant`**`(`*data*`)`

Return the number of irrelevant records.

**Parameters**

**`data`** (`asreview.Dataset`) – An Dataset object with the records.

**Returns**

*int* – The statistic

### asreview.data.statistics.n_keywords

`asreview.data.statistics.`**`n_keywords`**`(`*data*`)`

Return the number of keywords.

**Parameters**

**`data`** (`asreview.Dataset`) – An Dataset object with the records.

**Returns**

*int* – The statistic

### asreview.data.statistics.n_missing_abstract

`asreview.data.statistics.`**`n_missing_abstract`**`(`*data*`)`

Return the number of records with missing abstracts.

**Parameters**

**`data`** (`asreview.Dataset`) – An Dataset object with the records.

**Returns**

*int* – The statistic

### asreview.data.statistics.n_missing_title

`asreview.data.statistics.`**`n_missing_title`**(*data*)

Return the number of records with missing titles.

**Parameters**
**`data`** (`asreview.Dataset`) – An Dataset object with the records.

**Returns**
*int* – The statistic

### asreview.data.statistics.n_records

`asreview.data.statistics.`**`n_records`**(*data*)

Return the number of records.

**Parameters**
**`data`** (`asreview.Dataset`) – An Dataset object with the records.

**Returns**
*int* – The statistic

### asreview.data.statistics.n_relevant

`asreview.data.statistics.`**`n_relevant`**(*data*)

Return the number of relevant records.

**Parameters**
**`data`** (`asreview.Dataset`) – An Dataset object with the records.

**Returns**
*int* – The statistic

### asreview.data.statistics.n_unlabeled

`asreview.data.statistics.`**`n_unlabeled`**(*data*)

Return the number of unlabeled records.

**Parameters**
**`data`** (`asreview.Dataset`) – An Dataset object with the records.

**Returns**
*int* – The statistic

### asreview.data.statistics.title_length

`asreview.data.statistics.`**`title_length`**(*data*)

Return the average length of the titles.

**Parameters**
**`data`** (`asreview.Dataset`) – An Dataset object with the records.

**Returns**
*int* – The statistic

### 26.1.4 Datasets

#### Available datasets

| | |
|---|---|
| *asreview.datasets.SynergyDataGroup*() | Datasets available in the SYNERGY dataset. |
| *asreview.datasets.NaturePublicationDataGroup*() | Datasets used in the paper Van de Schoot et al. 2020. |

#### asreview.datasets.SynergyDataGroup

**class** asreview.datasets.**SynergyDataGroup**

Datasets available in the SYNERGY dataset.

##### Attributes

| |
|---|
| *description* |
| *group_id* |

##### asreview.datasets.SynergyDataGroup.description

SynergyDataGroup.**description = 'SYNERGY datasets (asreview.ai/synergy)'**

##### asreview.datasets.SynergyDataGroup.group_id

SynergyDataGroup.**group_id = 'synergy'**

##### Methods

| | |
|---|---|
| *append*(dataset) | Append dataset to group. |
| *find*(dataset_id) | Find dataset in the group. |

##### asreview.datasets.SynergyDataGroup.append

SynergyDataGroup.**append**(*dataset*)

Append dataset to group.

**dataset: asreview.datasets.BaseDataSet**
A asreview BaseDataSet-like object.

#### asreview.datasets.SynergyDataGroup.find

`SynergyDataGroup.find(dataset_id)`

Find dataset in the group.

**Parameters**

**dataset_id** (*str*) – Identifier of the dataset to look for. It can also be one of the aliases. Case insensitive.

**Returns**

*asreview.datasets.BaseDataSet* – Returns base dataset with the given dataset_id.

### asreview.datasets.NaturePublicationDataGroup

`class asreview.datasets.NaturePublicationDataGroup`

Datasets used in the paper Van de Schoot et al. 2020.

#### Attributes

| |
|---|
| *description* |
| *group_id* |

#### asreview.datasets.NaturePublicationDataGroup.description

`NaturePublicationDataGroup.description = 'Datasets used in the validation paper published in Nature Machine Intelligence (van de Schoot et al. 2021)'`

#### asreview.datasets.NaturePublicationDataGroup.group_id

`NaturePublicationDataGroup.group_id = 'benchmark-nature'`

#### Methods

| | |
|---|---|
| *append*(dataset) | Append dataset to group. |
| *find*(dataset_id) | Find dataset in the group. |

#### asreview.datasets.NaturePublicationDataGroup.append

`NaturePublicationDataGroup.append`(*dataset*)

Append dataset to group.

**dataset: asreview.datasets.BaseDataSet**
A asreview BaseDataSet-like object.

#### asreview.datasets.NaturePublicationDataGroup.find

`NaturePublicationDataGroup.find`(*dataset_id*)

Find dataset in the group.

**Parameters**
**`dataset_id`** (*str*) – Identifier of the dataset to look for. It can also be one of the aliases. Case insensitive.

**Returns**
*asreview.datasets.BaseDataSet* – Returns base dataset with the given dataset_id.

### Dataset managers

| |
|---|
| *asreview.datasets.BaseDataSet*(dataset_id[, ...]) |
| *asreview.datasets.BaseDataGroup*(*datasets) |
| *asreview.datasets.DatasetManager*() |

#### asreview.datasets.BaseDataSet

**class** `asreview.datasets.BaseDataSet`(*dataset_id*, *filepath=None*, *title=None*, *description=None*, *authors=None*, *topic=None*, *link=None*, *reference=None*, *img_url=None*, *license=None*, *year=None*, *aliases=None*, ***kwargs*)

#### Attributes

| |
|---|
| *filename* |
| *reader* |

#### asreview.datasets.BaseDataSet.filename

**property** BaseDataSet.**filename**

#### asreview.datasets.BaseDataSet.reader

**property** BaseDataSet.**reader**

#### Methods

| | |
|---|---|
| *to_file*(path) | |

#### asreview.datasets.BaseDataSet.to_file

BaseDataSet.**to_file**(*path*)

### asreview.datasets.BaseDataGroup

**class** asreview.datasets.**BaseDataGroup**(**datasets*)

#### Attributes

| | |
|---|---|
| *description* | |
| *group_id* | |

#### asreview.datasets.BaseDataGroup.description

**abstract property** BaseDataGroup.**description**

#### asreview.datasets.BaseDataGroup.group_id

**abstract property** BaseDataGroup.**group_id**

#### Methods

| | |
|---|---|
| *append*(dataset) | Append dataset to group. |
| *find*(dataset_id) | Find dataset in the group. |

#### asreview.datasets.BaseDataGroup.append

`BaseDataGroup.`**`append`**`(`*dataset*`)`

Append dataset to group.

**dataset: asreview.datasets.BaseDataSet**
A asreview BaseDataSet-like object.

#### asreview.datasets.BaseDataGroup.find

`BaseDataGroup.`**`find`**`(`*dataset_id*`)`

Find dataset in the group.

**Parameters**
**dataset_id** (*str*) – Identifier of the dataset to look for. It can also be one of the aliases. Case insensitive.

**Returns**
*asreview.datasets.BaseDataSet* – Returns base dataset with the given dataset_id.

### asreview.datasets.DatasetManager

**`class`** `asreview.datasets.`**`DatasetManager`**

#### Attributes

| |
|---|
| *groups* |

#### asreview.datasets.DatasetManager.groups

**`property`** `DatasetManager.`**`groups`**

### Methods

| | |
|---|---|
| *find*(dataset_id) | Find a dataset. |
| *list*([include, exclude, serialize, ...]) | List the available datasets. |

### asreview.datasets.DatasetManager.find

DatasetManager.**find**(*dataset_id*)

Find a dataset.

**Parameters**

**dataset_id** (*str, iterable*) – Look for this term in aliases within any dataset. A group can be specified by setting dataset_id to 'group_id:dataset_id'. This can be helpful if the dataset_id is not unique. The dataset_id can also be a non-string iterable, in which case a list will be returned with all terms. Dataset_ids should not contain semicolons (:). Return None if the dataset could not be found.

**Returns**

*BaseDataSet* – Return the dataset with dataset_id.

### asreview.datasets.DatasetManager.list

DatasetManager.**list**(*include=None*, *exclude=None*, *serialize=True*, *raise_on_error=False*)

List the available datasets.

**Parameters**

- **include** (*str, iterable*) – List of groups to include
- **exclude** (*str, iterable*) – List of groups to exclude from all groups.
- **serialize** (*bool*) – Make returned list serializable.
- **raise_on_error** (*bool*) – Raise error when entry point can't be loaded.

**Returns**

*list* – List with datasets as values.

## 26.2 Reviewer

| | |
|---|---|
| *simulation.Simulate*(as_data, project[, ...]) | ASReview Simulation mode class. |

### 26.2.1 asreview.simulation.Simulate

**class** asreview.simulation.**Simulate**(*as_data*, *project*, *classifier=<asreview.models.classifiers.nb.NaiveBayesClassifier object>*, *query_model=<asreview.models.query.max.MaxQuery object>*, *balance_model=<asreview.models.balance.simple.SimpleBalance object>*, *feature_model=<asreview.models.feature_extraction.tfidf.Tfidf object>*, *n_prior_included=0*, *n_prior_excluded=0*, *prior_indices=None*, *n_papers=None*, *n_instances=1*, *stop_if=None*, *start_idx=None*, *init_seed=None*, *write_interval=None*, ***kwargs*)

ASReview Simulation mode class.

**Parameters**

- **as_data** (asreview.Dataset) – The data object which contains the text, labels, etc.
- **model** (BaseModel) – Initialized model to fit the data during active learning. See asreview.models.utils.py for possible models.
- **query_model** (*BaseQueryModel*) – Initialized model to query new instances for review, such as random sampling or max sampling. See asreview.query_strategies.utils.py for query models.
- **balance_model** (*BaseBalanceModel*) – Initialized model to redistribute the training data during the active learning process. They might either resample or undersample specific papers.
- **feature_model** (*BaseFeatureModel*) – Feature extraction model that converts texts and keywords to feature matrices.
- **n_prior_included** (*int*) – Sample n prior included papers.
- **n_prior_excluded** (*int*) – Sample n prior excluded papers.
- **prior_indices** (*int*) – Prior indices by row number.
- **n_instances** (*int*) – Number of papers to query at each step in the active learning process.
- **stop_if** (*int*) – Number of steps/queries to perform. Set to None for no limit.
- **start_idx** (*numpy.ndarray*) – Start the simulation/review with these indices. They are assumed to be already labeled. Failing to do so might result bad behaviour.
- **init_seed** (*int*) – Seed for setting the prior indices if the –prior_idx option is not used. If the option prior_idx is used with one or more index, this option is ignored.
- **state_file** (*str*) – Path to state file.
- **write_interval** (*int*) – After how many labeled records to write the simulation data to the state.

#### Attributes

| | |
|---|---|
| *settings* | Get an ASReview settings object |

#### asreview.simulation.Simulate.settings

**property** Simulate.**settings**

Get an ASReview settings object

#### Methods

| | |
|---|---|
| *review*() | |
| *train*() | Train a new model on the labeled data. |

#### asreview.simulation.Simulate.review

Simulate.**review**()

#### asreview.simulation.Simulate.train

Simulate.**train**()

Train a new model on the labeled data.

## 26.3 Models

This section provides an overview of the available models for active learning in ASReview. For command line usage, use the name (`example`) given behind the model description (or see the name property of the model). Some models require additional dependencies, see the model class for more information and instructions. Base class

| | |
|---|---|
| *models.base.BaseModel*() | Abstract class for any kind of model. |

### 26.3.1 asreview.models.base.BaseModel

**class** asreview.models.base.**BaseModel**

Abstract class for any kind of model.

### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

#### asreview.models.base.BaseModel.default_param

**property** BaseModel.**default_param**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

#### asreview.models.base.BaseModel.name

BaseModel.**name** = **'base'**

#### asreview.models.base.BaseModel.param

**property** BaseModel.**param**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| |
|---|
| *full_hyper_space*() |
| *hyper_space*() |

#### asreview.models.base.BaseModel.full_hyper_space

BaseModel.**full_hyper_space**()

#### asreview.models.base.BaseModel.hyper_space

BaseModel.**hyper_space**()

## 26.3.2 asreview.models.feature_extraction

Classes

| | |
|---|---|
| *feature_extraction.base.BaseFeatureExtraction*([...]) | Base class for feature extraction methods. |
| *feature_extraction.Tfidf*(*args[, ngram_max, ...]) | TF-IDF feature extraction technique (`tfidf`). |
| *feature_extraction.Doc2Vec*(*args[, ...]) | Doc2Vec feature extraction technique (`doc2vec`). |
| *feature_extraction.EmbeddingIdf*(*args[, ...]) | Embedding IDF feature extraction technique (`embedding-idf`). |
| *feature_extraction.EmbeddingLSTM*(*args[, ...]) | Embedding LSTM feature extraction technique (`embedding-lstm`). |
| *feature_extraction.SBERT*(*args[, ...]) | Sentence BERT feature extraction technique (`sbert`). |

### asreview.models.feature_extraction.base.BaseFeatureExtraction

**class** asreview.models.feature_extraction.base.**BaseFeatureExtraction**(*split_ta=0*, *use_keywords=0*)

Base class for feature extraction methods.

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

#### asreview.models.feature_extraction.base.BaseFeatureExtraction.default_param

**property** BaseFeatureExtraction.**default_param**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.feature_extraction.base.BaseFeatureExtraction.name

`BaseFeatureExtraction.name = 'base-feature'`

### asreview.models.feature_extraction.base.BaseFeatureExtraction.param

**property** `BaseFeatureExtraction.param`

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

## Methods

| | |
|---|---|
| *fit*(texts) | Fit the model to the texts. |
| *fit_transform*(texts[, titles, abstracts, ...]) | Fit and transform a list of texts. |
| *full_hyper_space*() | |
| *hyper_space*() | |
| *transform*(texts) | Transform a list of texts. |

### asreview.models.feature_extraction.base.BaseFeatureExtraction.fit

`BaseFeatureExtraction.fit(`*texts*`)`

Fit the model to the texts.

It is not always necessary to implement this if there's not real fitting being done.

**Parameters**
**texts** (*numpy.ndarray*) – Texts to be fitted.

### asreview.models.feature_extraction.base.BaseFeatureExtraction.fit_transform

`BaseFeatureExtraction.fit_transform(`*texts*, *titles=None*, *abstracts=None*, *keywords=None*`)`

Fit and transform a list of texts.

**Parameters**
**texts** (*numpy.ndarray*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**
*numpy.ndarray* – Feature matrix representing the texts.

#### asreview.models.feature_extraction.base.BaseFeatureExtraction.full_hyper_space

`BaseFeatureExtraction.`**`full_hyper_space`**`()`

#### asreview.models.feature_extraction.base.BaseFeatureExtraction.hyper_space

`BaseFeatureExtraction.`**`hyper_space`**`()`

#### asreview.models.feature_extraction.base.BaseFeatureExtraction.transform

**`abstract`** `BaseFeatureExtraction.`**`transform`**`(`*texts*`)`

Transform a list of texts.

**Parameters**

**`texts`** (*`numpy.ndarray`*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**

*numpy.ndarray* – Feature matrix representing the texts.

### asreview.models.feature_extraction.Tfidf

**`class`** `asreview.models.feature_extraction.`**`Tfidf`**`(`**args*, *ngram_max=1*, *stop_words='english'*, ***kwargs*`)`

TF-IDF feature extraction technique (`tfidf`).

Use the standard TF-IDF (Term Frequency-Inverse Document Frequency) feature extraction technique from SKLearn. Gives a sparse matrix as output. Works well in combination with *`asreview.models.classifiers.NaiveBayesClassifier`* and other fast training models (given that the features vectors are relatively wide).

**Parameters**

- **`ngram_max`** (*`int`*) – Can use up to ngrams up to ngram_max. For example in the case of ngram_max=2, monograms and bigrams could be used.
- **`stop_words`** (*`str`*) – When set to 'english', use stopwords. If set to None or 'none', do not use stop words.

#### Attributes

| | |
|---|---|
| *`default_param`* | Get the default parameters of the model. |
| *`label`* | |
| *`name`* | |
| *`param`* | Get the (assigned) parameters of the model. |

### asreview.models.feature_extraction.Tfidf.default_param

**property** `Tfidf.default_param`

Get the default parameters of the model.

**Returns**
*dict* – Dictionary with parameter: default value

### asreview.models.feature_extraction.Tfidf.label

`Tfidf.label = 'TF-IDF'`

### asreview.models.feature_extraction.Tfidf.name

`Tfidf.name = 'tfidf'`

### asreview.models.feature_extraction.Tfidf.param

**property** `Tfidf.param`

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *fit*(texts) | Fit the model to the texts. |
| *fit_transform*(texts[, titles, abstracts, ...]) | Fit and transform a list of texts. |
| *full_hyper_space*() | |
| *hyper_space*() | |
| *transform*(texts) | Transform a list of texts. |

### asreview.models.feature_extraction.Tfidf.fit

`Tfidf.fit(`*texts*`)`

Fit the model to the texts.

It is not always necessary to implement this if there's not real fitting being done.

**Parameters**
**texts** (*numpy.ndarray*) – Texts to be fitted.

#### asreview.models.feature_extraction.Tfidf.fit_transform

`Tfidf.`**`fit_transform`**`(`*texts*, *titles=None*, *abstracts=None*, *keywords=None*`)`

Fit and transform a list of texts.

**Parameters**
**`texts`** (*`numpy.ndarray`*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**
*numpy.ndarray* – Feature matrix representing the texts.

#### asreview.models.feature_extraction.Tfidf.full_hyper_space

`Tfidf.`**`full_hyper_space`**`()`

#### asreview.models.feature_extraction.Tfidf.hyper_space

`Tfidf.`**`hyper_space`**`()`

#### asreview.models.feature_extraction.Tfidf.transform

`Tfidf.`**`transform`**`(`*texts*`)`

Transform a list of texts.

**Parameters**
**`texts`** (*`numpy.ndarray`*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**
*numpy.ndarray* – Feature matrix representing the texts.

### asreview.models.feature_extraction.Doc2Vec

`class asreview.models.feature_extraction.`**`Doc2Vec`**`(`**args*, *vector_size=40*, *epochs=33*, *min_count=1*, *n_jobs=1*, *window=7*, *dm_concat=0*, *dm=2*, *dbow_words=0*, ***kwargs*`)`

Doc2Vec feature extraction technique (`doc2vec`).

Feature extraction technique provided by the gensim package. It takes relatively long to create a feature matrix with this method. However, this only has to be done once per simulation/review. The upside of this method is the dimension- reduction that generally takes place, which makes the modelling quicker.

**Note:** This feature extraction technique requires `gensim` to be installed. Use `pip install asreview[gensim]` or install all optional ASReview dependencies with `pip install asreview[all]`

**Parameters**

- **`vector_size`** (*`int`*) – Output size of the vector.
- **`epochs`** (*`int`*) – Number of epochs to train the doc2vec model.

- **min_count** (*int*) – Minimum number of occurences for a word in the corpus for it to be included in the model.
- **n_jobs** (*int*) – Number of threads to train the model with.
- **window** (*int*) – Maximum distance over which word vectors influence each other.
- **dm_concat** (*int*) – Whether to concatenate word vectors or not. See paper for more detail.
- **dm** (*int*) – Model to use. 0: Use distribute bag of words (DBOW). 1: Use distributed memory (DM). 2: Use both of the above with half the vector size and concatenate them.
- **dbow_words** (*int*) – Whether to train the word vectors using the skipgram method.

### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.feature_extraction.Doc2Vec.default_param

**property** Doc2Vec.**default_param**

Get the default parameters of the model.

**Returns**
*dict* – Dictionary with parameter: default value

### asreview.models.feature_extraction.Doc2Vec.label

Doc2Vec.**label** = **'Doc2Vec'**

### asreview.models.feature_extraction.Doc2Vec.name

Doc2Vec.**name** = **'doc2vec'**

### asreview.models.feature_extraction.Doc2Vec.param

**property** Doc2Vec.**param**

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *fit*(texts) | Fit the model to the texts. |
| *fit_transform*(texts[, titles, abstracts, ...]) | Fit and transform a list of texts. |
| *full_hyper_space*() | |
| *hyper_space*() | |
| *transform*(texts) | Transform a list of texts. |

#### asreview.models.feature_extraction.Doc2Vec.fit

Doc2Vec.**fit**(*texts*)

Fit the model to the texts.

It is not always necessary to implement this if there's not real fitting being done.

**Parameters**
`texts` (*numpy.ndarray*) – Texts to be fitted.

#### asreview.models.feature_extraction.Doc2Vec.fit_transform

Doc2Vec.**fit_transform**(*texts*, *titles=None*, *abstracts=None*, *keywords=None*)

Fit and transform a list of texts.

**Parameters**
`texts` (*numpy.ndarray*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**
*numpy.ndarray* – Feature matrix representing the texts.

#### asreview.models.feature_extraction.Doc2Vec.full_hyper_space

Doc2Vec.**full_hyper_space**()

#### asreview.models.feature_extraction.Doc2Vec.hyper_space

Doc2Vec.**hyper_space**()

#### asreview.models.feature_extraction.Doc2Vec.transform

Doc2Vec.**transform**(*texts*)

Transform a list of texts.

**Parameters**
`texts` (*numpy.ndarray*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**
*numpy.ndarray* – Feature matrix representing the texts.

### asreview.models.feature_extraction.EmbeddingIdf

**class** `asreview.models.feature_extraction.`**`EmbeddingIdf`**(**args*, *embedding_fp=None*, *random_state=None*, ***kwargs*)

Embedding IDF feature extraction technique (`embedding-idf`).

This model averages the weighted word vectors of all the words in the text, in order to get a single feature vector for each text. The weights are provided by the inverse document frequencies.

**Note:** This feature extraction technique requires `tensorflow` to be installed. Use `pip install asreview[tensorflow]` or install all optional ASReview dependencies with `pip install asreview[all]`

**Parameters**
**`embedding_fp`** (*str*) – Path to embedding.

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

#### asreview.models.feature_extraction.EmbeddingIdf.default_param

**property** `EmbeddingIdf.`**`default_param`**

Get the default parameters of the model.

**Returns**
*dict* – Dictionary with parameter: default value

#### asreview.models.feature_extraction.EmbeddingIdf.label

`EmbeddingIdf.`**`label = 'Embedding IDF'`**

#### asreview.models.feature_extraction.EmbeddingIdf.name

`EmbeddingIdf.`**`name = 'embedding-idf'`**

### asreview.models.feature_extraction.EmbeddingIdf.param

`property EmbeddingIdf.param`

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *fit*(texts) | Fit the model to the texts. |
| *fit_transform*(texts[, titles, abstracts, ...]) | Fit and transform a list of texts. |
| *full_hyper_space*() | |
| *hyper_space*() | |
| *transform*(texts) | Transform a list of texts. |

### asreview.models.feature_extraction.EmbeddingIdf.fit

`EmbeddingIdf.fit(texts)`

Fit the model to the texts.

It is not always necessary to implement this if there's not real fitting being done.

**Parameters**

**texts** (*numpy.ndarray*) – Texts to be fitted.

### asreview.models.feature_extraction.EmbeddingIdf.fit_transform

`EmbeddingIdf.fit_transform(texts, titles=None, abstracts=None, keywords=None)`

Fit and transform a list of texts.

**Parameters**

**texts** (*numpy.ndarray*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**

*numpy.ndarray* – Feature matrix representing the texts.

### asreview.models.feature_extraction.EmbeddingIdf.full_hyper_space

`EmbeddingIdf.full_hyper_space()`

### asreview.models.feature_extraction.EmbeddingIdf.hyper_space

`EmbeddingIdf.`**`hyper_space`**`()`

### asreview.models.feature_extraction.EmbeddingIdf.transform

`EmbeddingIdf.`**`transform`**`(`*texts*`)`

Transform a list of texts.

**Parameters**

**`texts`** (*`numpy.ndarray`*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**

*numpy.ndarray* – Feature matrix representing the texts.

## asreview.models.feature_extraction.EmbeddingLSTM

**`class`** `asreview.models.feature_extraction.`**`EmbeddingLSTM`**`(`**args*, *loop_sequence=1*, *num_words=20000*, *max_sequence_length=1000*, *padding='post'*, *truncating='post'*, *n_jobs=1*, ***kwargs*`)`

Embedding LSTM feature extraction technique (`embedding-lstm`).

Feature extraction technique for `asreview.models.classifiers.LSTMBaseClassifier` and `asreview.models.classifiers.LSTMPoolClassifier` models.

**Note:** This feature extraction technique requires `tensorflow` to be installed. Use `pip install asreview[tensorflow]` or install all optional ASReview dependencies with `pip install asreview[all]`

**Parameters**

- **`loop_sequence`** (*`bool`*) – Instead of zeros at the start/end of sequence loop it.
- **`num_words`** (*`int`*) – Maximum number of unique words to be processed.
- **`max_sequence_length`** (*`int`*) – Maximum length of the sequence. Shorter get struncated. Longer sequences get either padded with zeros or looped.
- **`padding`** (*`str`*) – Which side should be padded [pre/post].
- **`truncating`** – Which side should be truncated [pre/post].
- **`n_jobs`** – Number of processors used in reading the embedding matrix.

### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

#### asreview.models.feature_extraction.EmbeddingLSTM.default_param

**property** EmbeddingLSTM.**default_param**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

#### asreview.models.feature_extraction.EmbeddingLSTM.label

`EmbeddingLSTM.label = 'Embedding LSTM'`

#### asreview.models.feature_extraction.EmbeddingLSTM.name

`EmbeddingLSTM.name = 'embedding-lstm'`

#### asreview.models.feature_extraction.EmbeddingLSTM.param

**property** EmbeddingLSTM.**param**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *fit*(texts) | Fit the model to the texts. |
| *fit_transform*(texts[, titles, abstracts, ...]) | Fit and transform a list of texts. |
| *full_hyper_space*() | |
| *get_embedding_matrix*(texts, embedding_fp) | |
| *hyper_space*() | |
| *transform*(texts) | Transform a list of texts. |

### asreview.models.feature_extraction.EmbeddingLSTM.fit

`EmbeddingLSTM.`**`fit`**`(`*texts*`)`

Fit the model to the texts.

It is not always necessary to implement this if there's not real fitting being done.

**Parameters**

**`texts`** (*`numpy.ndarray`*) – Texts to be fitted.

### asreview.models.feature_extraction.EmbeddingLSTM.fit_transform

`EmbeddingLSTM.`**`fit_transform`**`(`*texts*, *titles=None*, *abstracts=None*, *keywords=None*`)`

Fit and transform a list of texts.

**Parameters**

**`texts`** (*`numpy.ndarray`*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**

*numpy.ndarray* – Feature matrix representing the texts.

### asreview.models.feature_extraction.EmbeddingLSTM.full_hyper_space

`EmbeddingLSTM.`**`full_hyper_space`**`()`

### asreview.models.feature_extraction.EmbeddingLSTM.get_embedding_matrix

`EmbeddingLSTM.`**`get_embedding_matrix`**`(`*texts*, *embedding_fp*`)`

### asreview.models.feature_extraction.EmbeddingLSTM.hyper_space

`EmbeddingLSTM.`**`hyper_space`**`()`

### asreview.models.feature_extraction.EmbeddingLSTM.transform

`EmbeddingLSTM.`**`transform`**`(`*texts*`)`

Transform a list of texts.

**Parameters**

**`texts`** (*`numpy.ndarray`*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**

*numpy.ndarray* – Feature matrix representing the texts.

### asreview.models.feature_extraction.SBERT

**class** asreview.models.feature_extraction.**SBERT**(**args*, *transformer_model='all-mpnet-base-v2'*, *is_pretrained_sbert=True*, *pooling_mode='mean'*, ***kwargs*)

Sentence BERT feature extraction technique (`sbert`).

By setting the `transformer_model` parameter, you can use other transformer models. For example, `transformer_model='bert-base-nli-stsb- large'`. For a list of available models, see the Sentence BERT documentation.

Sentence BERT is a sentence embedding model that is trained on a large corpus of human written text. It is a fast and accurate model that can be used for many tasks.

The huggingface library includes multilingual text classification models. If your dataset contains records with multiple languages, you can use the `transformer_model` parameter to select the model that is most suitable for your data.

**Note:** This feature extraction technique requires `sentence_transformers` to be installed. Use `pip install asreview[sentence_transformers]` or install all optional ASReview dependencies with `pip install asreview[all]` to install the package.

**Parameters**

- **transformer_model** (*str, optional*) – The transformer model to use. Default: 'all-mpnet-base-v2'
- **is_pretrained_SBERT** (*boolean, optional*) – Default: True
- **pooling_mode** (*str, optional*) – Pooling mode to get sentence embeddings from word embeddings Default: 'mean' Other options available are 'mean', 'max' and 'cls'. Only used if is_pretrained_SBERT=False mean: Uses mean pooling of word embeddings max: Uses max pooling of word embeddings cls: Uses embeddings of [CLS] token as sentence embeddings

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.feature_extraction.SBERT.default_param

**property** SBERT.**default_param**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.feature_extraction.SBERT.label

SBERT.**label** = **'Sentence BERT'**

### asreview.models.feature_extraction.SBERT.name

SBERT.**name** = **'sbert'**

### asreview.models.feature_extraction.SBERT.param

**property** SBERT.**param**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *fit*(texts) | Fit the model to the texts. |
| *fit_transform*(texts[, titles, abstracts, ...]) | Fit and transform a list of texts. |
| *full_hyper_space*() | |
| *hyper_space*() | |
| *transform*(texts) | Transform a list of texts. |

### asreview.models.feature_extraction.SBERT.fit

SBERT.**fit**(*texts*)

Fit the model to the texts.

It is not always necessary to implement this if there's not real fitting being done.

**Parameters**

**texts** (*numpy.ndarray*) – Texts to be fitted.

### asreview.models.feature_extraction.SBERT.fit_transform

SBERT.**fit_transform**(*texts*, *titles=None*, *abstracts=None*, *keywords=None*)

Fit and transform a list of texts.

**Parameters**
**texts** (*numpy.ndarray*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**
*numpy.ndarray* – Feature matrix representing the texts.

### asreview.models.feature_extraction.SBERT.full_hyper_space

SBERT.**full_hyper_space**()

### asreview.models.feature_extraction.SBERT.hyper_space

SBERT.**hyper_space**()

### asreview.models.feature_extraction.SBERT.transform

SBERT.**transform**(*texts*)

Transform a list of texts.

**Parameters**
**texts** (*numpy.ndarray*) – A sequence of texts to be transformed. They are not yet tokenized.

**Returns**
*numpy.ndarray* – Feature matrix representing the texts.

Functions

| | |
|---|---|
| *feature_extraction.get_feature_model*(name, *args) | Get an instance of a feature extraction model from a string. |
| *feature_extraction.get_feature_class*(name) | Get class of feature extraction from string. |
| *feature_extraction.list_feature_extraction*() | List available feature extraction method classes. |

### asreview.models.feature_extraction.get_feature_model

asreview.models.feature_extraction.**get_feature_model**(*name*, **args*, *random_state=None*, ***kwargs*)

Get an instance of a feature extraction model from a string.

**Parameters**

- **name** (*str*) – Name of the feature extraction model.
- ***args** – Arguments for the feature extraction model.
- ****kwargs** – Keyword arguments for thefeature extraction model.

**Returns**
*BaseFeatureExtraction* – Initialized instance of feature extraction algorithm.

#### asreview.models.feature_extraction.get_feature_class

`asreview.models.feature_extraction.`**`get_feature_class`**`(`*`name`*`)`

Get class of feature extraction from string.

**Parameters**

**`name`** (*`str`*) – Name of the feature model, e.g. 'doc2vec', 'tfidf' or 'embedding-lstm'.

**Returns**

*BaseFeatureExtraction* – Class corresponding to the name.

#### asreview.models.feature_extraction.list_feature_extraction

`asreview.models.feature_extraction.`**`list_feature_extraction`**`()`

List available feature extraction method classes.

**Returns**

*list* – Classes of available feature extraction methods in alphabetical order.

### 26.3.3 `asreview.models.classifiers`

Classes

| | |
|---|---|
| *`classifiers.base.BaseTrainClassifier`*() | Base model, abstract class to be implemented by derived ones. |
| *`classifiers.NaiveBayesClassifier`*([alpha]) | Naive Bayes classifier (`nb`). |
| *`classifiers.RandomForestClassifier`*([...]) | Random forest classifier (`rf`). |
| *`classifiers.SVMClassifier`*([gamma, ...]) | Support vector machine classifier (`svm`). |
| *`classifiers.LogisticClassifier`*([C, ...]) | Logistic regression classifier (`logistic`). |
| *`classifiers.NN2LayerClassifier`*([...]) | Fully connected neural network (2 hidden layers) classifier (`nn-2-layer`). |

#### asreview.models.classifiers.base.BaseTrainClassifier

`class asreview.models.classifiers.base.`**`BaseTrainClassifier`**

Base model, abstract class to be implemented by derived ones.

All the non-abstract methods are okay if they are not implemented. There is a distinction between model parameters, which are needed during model creation and fit parameters, which are used during the fitting process. Fit parameters can be distinct from fit_kwargs (which are passed to the fit function).

#### Attributes

| | |
|---|---|
| *`default_param`* | Get the default parameters of the model. |
| *`name`* | |
| *`param`* | Get the (assigned) parameters of the model. |

### asreview.models.classifiers.base.BaseTrainClassifier.default_param

**property** `BaseTrainClassifier.`**`default_param`**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.classifiers.base.BaseTrainClassifier.name

`BaseTrainClassifier.`**`name`** `=` **`'base-train'`**

### asreview.models.classifiers.base.BaseTrainClassifier.param

**property** `BaseTrainClassifier.`**`param`**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

## Methods

| | |
|---|---|
| *`fit`*(X, y) | Fit the model to the data. |
| *`full_hyper_space`*() | |
| *`hyper_space`*() | |
| *`predict_proba`*(X) | Get the inclusion probability for each sample. |

### asreview.models.classifiers.base.BaseTrainClassifier.fit

`BaseTrainClassifier.`**`fit`**`(`*X*, *y*`)`

Fit the model to the data.

**Parameters**

- **X** (*`numpy.ndarray`*) – Feature matrix to fit.
- **y** (*`numpy.ndarray`*) – Labels for supervised learning.

#### asreview.models.classifiers.base.BaseTrainClassifier.full_hyper_space

BaseTrainClassifier.**full_hyper_space**()

#### asreview.models.classifiers.base.BaseTrainClassifier.hyper_space

BaseTrainClassifier.**hyper_space**()

#### asreview.models.classifiers.base.BaseTrainClassifier.predict_proba

BaseTrainClassifier.**predict_proba**(*X*)

Get the inclusion probability for each sample.

**Parameters**

**X** (*numpy.ndarray*) – Feature matrix to predict.

**Returns**

*numpy.ndarray* – Array with the probabilities for each class, with two columns (class 0, and class 1) and the number of samples rows.

### asreview.models.classifiers.NaiveBayesClassifier

**class** asreview.models.classifiers.**NaiveBayesClassifier**(*alpha=3.822*)

Naive Bayes classifier (nb).

Naive Bayes classifier. Only works in combination with the *asreview.models.feature_extraction.Tfidf* feature extraction model. Though relatively simplistic, seems to work quite well on a wide range of datasets.

The naive Bayes classifier is an implementation based on the sklearn multinomial naive Bayes classifier.

**Parameters**

**alpha** (*float, default=3.822*) – Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing).

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.classifiers.NaiveBayesClassifier.default_param

**property** NaiveBayesClassifier.**default_param**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.classifiers.NaiveBayesClassifier.label

NaiveBayesClassifier.**label** = **'Naive Bayes'**

### asreview.models.classifiers.NaiveBayesClassifier.name

NaiveBayesClassifier.**name** = **'nb'**

### asreview.models.classifiers.NaiveBayesClassifier.param

**property** NaiveBayesClassifier.**param**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *fit*(X, y) | Fit the model to the data. |
| *full_hyper_space*() | |
| *hyper_space*() | |
| *predict_proba*(X) | Get the inclusion probability for each sample. |

### asreview.models.classifiers.NaiveBayesClassifier.fit

NaiveBayesClassifier.**fit**(*X*, *y*)

Fit the model to the data.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix to fit.
- **y** (*numpy.ndarray*) – Labels for supervised learning.

#### asreview.models.classifiers.NaiveBayesClassifier.full_hyper_space

NaiveBayesClassifier.**full_hyper_space**()

#### asreview.models.classifiers.NaiveBayesClassifier.hyper_space

NaiveBayesClassifier.**hyper_space**()

#### asreview.models.classifiers.NaiveBayesClassifier.predict_proba

NaiveBayesClassifier.**predict_proba**(*X*)

Get the inclusion probability for each sample.

**Parameters**

**X** (*numpy.ndarray*) – Feature matrix to predict.

**Returns**

*numpy.ndarray* – Array with the probabilities for each class, with two columns (class 0, and class 1) and the number of samples rows.

### asreview.models.classifiers.RandomForestClassifier

class asreview.models.classifiers.**RandomForestClassifier**(*n_estimators=100*, *max_features=10*, *class_weight=1.0*, *random_state=None*)

Random forest classifier (rf).

The Random Forest classifier is an implementation based on the sklearn Random Forest classifier.

**Parameters**

- **n_estimators** (*int, default=100*) – The number of trees in the forest.
- **max_features** (*int, default=10*) – Number of features in the model.
- **class_weight** (*float, default=1.0*) – Class weight of the inclusions.
- **random_state** (*int or asreview.utils.SeededRandomState, default=None*) – Controls both the randomness of the bootstrapping of the samples used when building trees and the sampling of the features to consider when looking for the best split at each node.

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.classifiers.RandomForestClassifier.default_param

**property** `RandomForestClassifier.`**`default_param`**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.classifiers.RandomForestClassifier.label

`RandomForestClassifier.`**`label = 'Random forest'`**

### asreview.models.classifiers.RandomForestClassifier.name

`RandomForestClassifier.`**`name = 'rf'`**

### asreview.models.classifiers.RandomForestClassifier.param

**property** `RandomForestClassifier.`**`param`**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *fit*(X, y) | Fit the model to the data. |
| *full_hyper_space*() | |
| *hyper_space*() | |
| *predict_proba*(X) | Get the inclusion probability for each sample. |

### asreview.models.classifiers.RandomForestClassifier.fit

`RandomForestClassifier.`**`fit`**`(X, y)`

Fit the model to the data.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix to fit.
- **y** (*numpy.ndarray*) – Labels for supervised learning.

#### asreview.models.classifiers.RandomForestClassifier.full_hyper_space

`RandomForestClassifier.full_hyper_space()`

#### asreview.models.classifiers.RandomForestClassifier.hyper_space

`RandomForestClassifier.hyper_space()`

#### asreview.models.classifiers.RandomForestClassifier.predict_proba

`RandomForestClassifier.predict_proba(X)`

Get the inclusion probability for each sample.

**Parameters**

**X** (*numpy.ndarray*) – Feature matrix to predict.

**Returns**

*numpy.ndarray* – Array with the probabilities for each class, with two columns (class 0, and class 1) and the number of samples rows.

### asreview.models.classifiers.SVMClassifier

`class asreview.models.classifiers.SVMClassifier(gamma='auto', class_weight=0.249, C=15.4, kernel='linear', random_state=None)`

Support vector machine classifier (svm).

The Support Vector Machine classifier is an implementation based on the sklearn Support Vector Machine classifier.

**Parameters**

- **gamma** (*str*) – Gamma parameter of the SVM model.
- **class_weight** (*float*) – class_weight of the inclusions.
- **C** (*float*) – C parameter of the SVM model.
- **kernel** (*str*) – SVM kernel type.
- **random_state** (*int, asreview.utils.SeededRandomState*) – State of the RNG.

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.classifiers.SVMClassifier.default_param

**property** SVMClassifier.**default_param**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.classifiers.SVMClassifier.label

SVMClassifier.**label = 'Support vector machine'**

### asreview.models.classifiers.SVMClassifier.name

SVMClassifier.**name = 'svm'**

### asreview.models.classifiers.SVMClassifier.param

**property** SVMClassifier.**param**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *fit*(X, y) | Fit the model to the data. |
| *full_hyper_space*() | |
| *hyper_space*() | |
| *predict_proba*(X) | Get the inclusion probability for each sample. |

### asreview.models.classifiers.SVMClassifier.fit

SVMClassifier.**fit**(*X*, *y*)

Fit the model to the data.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix to fit.
- **y** (*numpy.ndarray*) – Labels for supervised learning.

#### asreview.models.classifiers.SVMClassifier.full_hyper_space

`SVMClassifier.`**`full_hyper_space`**`()`

#### asreview.models.classifiers.SVMClassifier.hyper_space

`SVMClassifier.`**`hyper_space`**`()`

#### asreview.models.classifiers.SVMClassifier.predict_proba

`SVMClassifier.`**`predict_proba`**`(`*X*`)`

Get the inclusion probability for each sample.

**Parameters**

**X** (*numpy.ndarray*) – Feature matrix to predict.

**Returns**

*numpy.ndarray* – Array with the probabilities for each class, with two columns (class 0, and class 1) and the number of samples rows.

### asreview.models.classifiers.LogisticClassifier

**class** asreview.models.classifiers.**LogisticClassifier**(*C=1.0*, *class_weight=1.0*, *random_state=None*, *n_jobs=1*)

Logistic regression classifier (`logistic`).

The Logistic regressions classifier is an implementation based on the sklearn Logistic regressions classifier.

**Parameters**

- **C** (*float*) – Parameter inverse to the regularization strength of the model.
- **class_weight** (*float*) – Class weight of the inclusions.
- **random_state** (*int, asreview.utils.SeededRandomState*) – Random state for the model.
- **n_jobs** (*int*) – Number of CPU cores used.

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.classifiers.LogisticClassifier.default_param

**property** LogisticClassifier.**default_param**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.classifiers.LogisticClassifier.label

LogisticClassifier.**label = 'Logistic regression'**

### asreview.models.classifiers.LogisticClassifier.name

LogisticClassifier.**name = 'logistic'**

### asreview.models.classifiers.LogisticClassifier.param

**property** LogisticClassifier.**param**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *fit*(X, y) | Fit the model to the data. |
| *full_hyper_space*() | |
| *hyper_space*() | |
| *predict_proba*(X) | Get the inclusion probability for each sample. |

### asreview.models.classifiers.LogisticClassifier.fit

LogisticClassifier.**fit**(*X*, *y*)

Fit the model to the data.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix to fit.
- **y** (*numpy.ndarray*) – Labels for supervised learning.

#### asreview.models.classifiers.LogisticClassifier.full_hyper_space

`LogisticClassifier.`**`full_hyper_space`**`()`

#### asreview.models.classifiers.LogisticClassifier.hyper_space

`LogisticClassifier.`**`hyper_space`**`()`

#### asreview.models.classifiers.LogisticClassifier.predict_proba

`LogisticClassifier.`**`predict_proba`**`(`*X*`)`

Get the inclusion probability for each sample.

**Parameters**

**X** (*numpy.ndarray*) – Feature matrix to predict.

**Returns**

*numpy.ndarray* – Array with the probabilities for each class, with two columns (class 0, and class 1) and the number of samples rows.

### asreview.models.classifiers.NN2LayerClassifier

**class** `asreview.models.classifiers.`**`NN2LayerClassifier`**`(`*dense_width=128*, *optimizer='rmsprop'*, *learn_rate=1.0*, *regularization=0.01*, *verbose=0*, *epochs=35*, *batch_size=32*, *shuffle=False*, *class_weight=30.0*`)`

Fully connected neural network (2 hidden layers) classifier (`nn-2-layer`).

Neural network with two hidden, dense layers of the same size.

Recommended feature extraction model is *asreview.models.feature_extraction.Doc2Vec*.

**Note:** This model requires `tensorflow` to be installed. Use `pip install asreview[tensorflow]` or install all optional ASReview dependencies with `pip install asreview[all]`

**Warning:** Might crash on some systems with limited memory in combination with *asreview.models.feature_extraction.Tfidf*.

**Parameters**

- **dense_width** (*int*) – Size of the dense layers.
- **optimizer** (*str*) – Name of the Keras optimizer.
- **learn_rate** (*float*) – Learning rate multiplier of the default learning rate.
- **regularization** (*float*) – Strength of the regularization on the weights and biases.
- **verbose** (*int*) – Verbosity of the model mirroring the values for Keras.
- **epochs** (*int*) – Number of epochs to train the neural network.
- **batch_size** (*int*) – Batch size used for the neural network.

- **shuffle** (*bool*) – Whether to shuffle the training data prior to training.
- **class_weight** (*float*) – Class weights for inclusions (1's).

### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

#### asreview.models.classifiers.NN2LayerClassifier.default_param

**property** NN2LayerClassifier.**default_param**

Get the default parameters of the model.

**Returns**
*dict* – Dictionary with parameter: default value

#### asreview.models.classifiers.NN2LayerClassifier.label

NN2LayerClassifier.**label** = **'Fully connected neural network (2 hidden layers)'**

#### asreview.models.classifiers.NN2LayerClassifier.name

NN2LayerClassifier.**name** = **'nn-2-layer'**

#### asreview.models.classifiers.NN2LayerClassifier.param

**property** NN2LayerClassifier.**param**

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *fit*(X, y) | Fit the model to the data. |
| *full_hyper_space*() | |
| *hyper_space*() | |
| *predict_proba*(X) | Get the inclusion probability for each sample. |

### asreview.models.classifiers.NN2LayerClassifier.fit

`NN2LayerClassifier.fit(X, y)`

Fit the model to the data.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix to fit.
- **y** (*numpy.ndarray*) – Labels for supervised learning.

### asreview.models.classifiers.NN2LayerClassifier.full_hyper_space

`NN2LayerClassifier.full_hyper_space()`

### asreview.models.classifiers.NN2LayerClassifier.hyper_space

`NN2LayerClassifier.hyper_space()`

### asreview.models.classifiers.NN2LayerClassifier.predict_proba

`NN2LayerClassifier.predict_proba(X)`

Get the inclusion probability for each sample.

**Parameters**

**X** (*numpy.ndarray*) – Feature matrix to predict.

**Returns**

*numpy.ndarray* – Array with the probabilities for each class, with two columns (class 0, and class 1) and the number of samples rows.

Functions

| | |
|---|---|
| *classifiers.get_classifier*(name, *args[, ...]) | Get an instance of a model from a string. |
| *classifiers.get_classifier_class*(name) | Get class of model from string. |
| *classifiers.list_classifiers*() | List available classifier classes. |

### asreview.models.classifiers.get_classifier

`asreview.models.classifiers.get_classifier(name, *args, random_state=None, **kwargs)`

Get an instance of a model from a string.

**Parameters**

- **name** (*str*) – Name of the model.
- ***args** – Arguments for the model.
- ****kwargs** – Keyword arguments for the model.

**Returns**

*BaseFeatureExtraction* – Initialized instance of classifier.

### asreview.models.classifiers.get_classifier_class

`asreview.models.classifiers.get_classifier_class(name)`

Get class of model from string.

**Parameters**

**name** (*str*) – Name of the model, e.g. 'svm', 'nb' or 'lstm-pool'.

**Returns**

*BaseModel* – Class corresponding to the name.

### asreview.models.classifiers.list_classifiers

`asreview.models.classifiers.list_classifiers()`

List available classifier classes.

**Returns**

*list* – Classes of available classifiers in alphabetical order.

## 26.3.4 asreview.models.query

Classes

| | |
|---|---|
| *query.base.BaseQueryStrategy*() | Abstract class for query strategies. |
| *query.base.ProbaQueryStrategy*() | |
| *query.MaxQuery*() | Maximum query strategy (`max`). |
| *query.MixedQuery*([strategy_1, strategy_2, ...]) | Mixed query strategy. |
| *query.MaxRandomQuery*([mix_ratio, random_state]) | Mixed (95% Maximum and 5% Random) query strategy (`max_random`). |
| *query.MaxUncertaintyQuery*([mix_ratio, ...]) | Mixed (95% Maximum and 5% Uncertainty) query strategy (`max_uncertainty`). |
| *query.UncertaintyQuery*() | Uncertainty query strategy (`uncertainty`). |
| *query.RandomQuery*([random_state]) | Random query strategy (`random`). |
| *query.ClusterQuery*([cluster_size, ...]) | Clustering query strategy (`cluster`). |

### asreview.models.query.base.BaseQueryStrategy

`class asreview.models.query.base.BaseQueryStrategy`

Abstract class for query strategies.

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.query.base.BaseQueryStrategy.default_param

**property** BaseQueryStrategy.**default_param**

Get the default parameters of the model.

**Returns**
*dict* – Dictionary with parameter: default value

### asreview.models.query.base.BaseQueryStrategy.name

BaseQueryStrategy.**name = 'base-query'**

### asreview.models.query.base.BaseQueryStrategy.param

**property** BaseQueryStrategy.**param**

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *query*(X[, classifier, n_instances, ...]) | Put records in ranked order. |

### asreview.models.query.base.BaseQueryStrategy.full_hyper_space

BaseQueryStrategy.**full_hyper_space**()

### asreview.models.query.base.BaseQueryStrategy.hyper_space

BaseQueryStrategy.**hyper_space**()

### asreview.models.query.base.BaseQueryStrategy.query

**abstract** BaseQueryStrategy.**query**(*X*, *classifier=None*, *n_instances=None*, *return_classifier_scores=False*, ***kwargs*)

Put records in ranked order.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix where every row contains the features of a record.
- **classifier** (*SKLearnModel*) – Trained classifier to compute relevance scores.

- **n_instances** (*int*) – Number of records to query. If None returns all records in ranked order.
- **return_classifier_score** (*bool*) – Return the relevance scores produced by the classifier.

**Returns**

*numpy.ndarray or (numpy.ndarray, np.ndarray)* – The QueryStrategy ranks the row numbers of the feature matrix. It returns an array of shape (n_instances,) containing the row indices in ranked order. If n_instances is None, returns all row numbers in ranked order. If n_instances is an integer, it only returns the top n_instances. If return_classifier_scores=True, also returns a second array with the same number of rows as the feature matrix, containing the relevance scores predicted by the classifier. If the classifier is not used, this will be None.

## asreview.models.query.base.ProbaQueryStrategy

`class asreview.models.query.base.ProbaQueryStrategy`

### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.query.base.ProbaQueryStrategy.default_param

`property ProbaQueryStrategy.default_param`

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.query.base.ProbaQueryStrategy.name

`ProbaQueryStrategy.name = 'proba'`

### asreview.models.query.base.ProbaQueryStrategy.param

`property ProbaQueryStrategy.param`

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

#### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *query*(X, classifier[, n_instances, ...]) | Query method for strategies which use class probabilities. |

#### asreview.models.query.base.ProbaQueryStrategy.full_hyper_space

ProbaQueryStrategy.**full_hyper_space**()

#### asreview.models.query.base.ProbaQueryStrategy.hyper_space

ProbaQueryStrategy.**hyper_space**()

#### asreview.models.query.base.ProbaQueryStrategy.query

ProbaQueryStrategy.**query**(*X*, *classifier*, *n_instances=None*, *return_classifier_scores=False*, ***kwargs*)

Query method for strategies which use class probabilities.

### asreview.models.query.MaxQuery

class asreview.models.query.**MaxQuery**

Maximum query strategy (`max`).

Choose the most likely samples to be included according to the model.

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.query.MaxQuery.default_param

**property** MaxQuery.**default_param**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.query.MaxQuery.label

MaxQuery.**label = 'Maximum'**

### asreview.models.query.MaxQuery.name

MaxQuery.**name = 'max'**

### asreview.models.query.MaxQuery.param

**property** MaxQuery.**param**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *query*(X, classifier[, n_instances, ...]) | Query method for strategies which use class probabilities. |

### asreview.models.query.MaxQuery.full_hyper_space

MaxQuery.**full_hyper_space**()

### asreview.models.query.MaxQuery.hyper_space

MaxQuery.**hyper_space**()

#### asreview.models.query.MaxQuery.query

`MaxQuery.`**`query`**(*X*, *classifier*, *n_instances=None*, *return_classifier_scores=False*, ***kwargs*)

Query method for strategies which use class probabilities.

### asreview.models.query.MixedQuery

**`class`** `asreview.models.query.`**`MixedQuery`**(*strategy_1='max'*, *strategy_2='random'*, *mix_ratio=0.95*, *random_state=None*, ***kwargs*)

Mixed query strategy.

Use two different query strategies at the same time with a ratio of one to the other. A mix of two query strategies is used. For example mixing max and random sampling with a mix ratio of 0.95 would mean that at each query 95% of the instances would be sampled with the max query strategy after which the remaining 5% would be sampled with the random query strategy. It would be called the *max_random* query strategy. Every combination of primitive query strategy is possible.

**Parameters**

- **`strategy_1`** (*`str`*) – Name of the first query strategy. Default 'max'.
- **`strategy_2`** (*`str`*) – Name of the second query strategy. Default 'random'
- **`mix_ratio`** (*`float`*) – Sampling from strategy_1 and strategy_2 according a Bernoulli distribution. E.g. for mix_ratio=0.95, this implies strategy_1 with probability 0.95 and strategy_2 with probability 0.05. Default 0.95.
- **`random_state`** (*`float`*) – Seed for the numpy random number generator.
- **`**kwargs`** (*`dict`*) – Keyword arguments for the two strategy. To specify which of the strategies the argument is for, prepend with the name of the query strategy and an underscore, e.g. 'max' for maximal sampling.

#### Attributes

| | |
|---|---|
| *`default_param`* | Get the default parameters of the model. |
| *`name`* | str(object='') -> str str(bytes_or_buffer[, encoding[, errors]]) -> str |
| *`param`* | Get the (assigned) parameters of the model. |

#### asreview.models.query.MixedQuery.default_param

**`property`** `MixedQuery.`**`default_param`**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.query.MixedQuery.name

**property** MixedQuery.**name**

str(object='') -> str str(bytes_or_buffer[, encoding[, errors]]) -> str

Create a new string object from the given object. If encoding or errors is specified, then the object must expose a data buffer that will be decoded using the given encoding and error handler. Otherwise, returns the result of object.__str__() (if defined) or repr(object). encoding defaults to sys.getdefaultencoding(). errors defaults to 'strict'.

### asreview.models.query.MixedQuery.param

**property** MixedQuery.**param**

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *query*(X, classifier[, n_instances, ...]) | Put records in ranked order. |

### asreview.models.query.MixedQuery.full_hyper_space

MixedQuery.**full_hyper_space**()

### asreview.models.query.MixedQuery.hyper_space

MixedQuery.**hyper_space**()

### asreview.models.query.MixedQuery.query

MixedQuery.**query**(*X*, *classifier*, *n_instances=None*, *return_classifier_scores=False*, ***kwargs*)

Put records in ranked order.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix where every row contains the features of a record.
- **classifier** (*SKLearnModel*) – Trained classifier to compute relevance scores.
- **n_instances** (*int*) – Number of records to query. If None returns all records in ranked order.
- **return_classifier_score** (*bool*) – Return the relevance scores produced by the classifier.

**Returns**

*numpy.ndarray or (numpy.ndarray, np.ndarray)* – The QueryStrategy ranks the row numbers of the feature matrix. It returns an array of shape (n_instances,) containing the row indices in ranked order. If n_instances is None, returns all row numbers in ranked order. If n_instances is an integer, it only returns the top n_instances. If return_classifier_scores=True, also returns a second array with the same number of rows as the feature matrix, containing the relevance scores predicted by the classifier. If the classifier is not used, this will be None.

### asreview.models.query.MaxRandomQuery

`class asreview.models.query.MaxRandomQuery(mix_ratio=0.95, random_state=None, **kwargs)`

Mixed (95% Maximum and 5% Random) query strategy (`max_random`).

A mix of maximum and random query strategies with a mix ratio of 0.95. At each query 95% of the instances would be sampled with the maximum query strategy after which the remaining 5% would be sampled with the random query strategy.

#### Attributes

| | |
|---|---|
| `default_param` | Get the default parameters of the model. |
| `label` | |
| `name` | |
| `param` | Get the (assigned) parameters of the model. |

#### asreview.models.query.MaxRandomQuery.default_param

`property MaxRandomQuery.default_param`

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

#### asreview.models.query.MaxRandomQuery.label

`MaxRandomQuery.label = 'Mixed (95% Maximum and 5% Random)'`

#### asreview.models.query.MaxRandomQuery.name

`MaxRandomQuery.name = 'max_random'`

### asreview.models.query.MaxRandomQuery.param

**property** `MaxRandomQuery.`**`param`**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *`full_hyper_space`*() | |
| *`hyper_space`*() | |
| *`query`*(X, classifier[, n_instances, ...]) | Put records in ranked order. |

### asreview.models.query.MaxRandomQuery.full_hyper_space

`MaxRandomQuery.`**`full_hyper_space`**`()`

### asreview.models.query.MaxRandomQuery.hyper_space

`MaxRandomQuery.`**`hyper_space`**`()`

### asreview.models.query.MaxRandomQuery.query

`MaxRandomQuery.`**`query`**(*X*, *classifier*, *n_instances=None*, *return_classifier_scores=False*, ***kwargs*)

Put records in ranked order.

**Parameters**

- **`X`** (*`numpy.ndarray`*) – Feature matrix where every row contains the features of a record.
- **`classifier`** (*`SKLearnModel`*) – Trained classifier to compute relevance scores.
- **`n_instances`** (*`int`*) – Number of records to query. If None returns all records in ranked order.
- **`return_classifier_score`** (*`bool`*) – Return the relevance scores produced by the classifier.

**Returns**

*numpy.ndarray or (numpy.ndarray, np.ndarray)* – The QueryStrategy ranks the row numbers of the feature matrix. It returns an array of shape (n_instances,) containing the row indices in ranked order. If n_instances is None, returns all row numbers in ranked order. If n_instances is an integer, it only returns the top n_instances. If return_classifier_scores=True, also returns a second array with the same number of rows as the feature matrix, containing the relevance scores predicted by the classifier. If the classifier is not used, this will be None.

### asreview.models.query.MaxUncertaintyQuery

**class** `asreview.models.query.`**`MaxUncertaintyQuery`**(*mix_ratio=0.95*, *random_state=None*, ***kwargs*)

Mixed (95% Maximum and 5% Uncertainty) query strategy (`max_uncertainty`).

A mix of maximum and random query strategies with a mix ratio of 0.95. At each query 95% of the instances would be sampled with the maximum query strategy after which the remaining 5% would be sampled with the uncertainty query strategy.

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

#### asreview.models.query.MaxUncertaintyQuery.default_param

**property** `MaxUncertaintyQuery.`**`default_param`**

Get the default parameters of the model.

**Returns**
*dict* – Dictionary with parameter: default value

#### asreview.models.query.MaxUncertaintyQuery.label

`MaxUncertaintyQuery.`**`label = 'Mixed (95% Maximum and 5% Uncertainty)'`**

#### asreview.models.query.MaxUncertaintyQuery.name

`MaxUncertaintyQuery.`**`name = 'max_uncertainty'`**

#### asreview.models.query.MaxUncertaintyQuery.param

**property** `MaxUncertaintyQuery.`**`param`**

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *query*(X, classifier[, n_instances, ...]) | Put records in ranked order. |

#### asreview.models.query.MaxUncertaintyQuery.full_hyper_space

MaxUncertaintyQuery.**full_hyper_space**()

#### asreview.models.query.MaxUncertaintyQuery.hyper_space

MaxUncertaintyQuery.**hyper_space**()

#### asreview.models.query.MaxUncertaintyQuery.query

MaxUncertaintyQuery.**query**(*X*, *classifier*, *n_instances=None*, *return_classifier_scores=False*, ***kwargs*)

Put records in ranked order.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix where every row contains the features of a record.
- **classifier** (*SKLearnModel*) – Trained classifier to compute relevance scores.
- **n_instances** (*int*) – Number of records to query. If None returns all records in ranked order.
- **return_classifier_score** (*bool*) – Return the relevance scores produced by the classifier.

**Returns**

*numpy.ndarray or (numpy.ndarray, np.ndarray)* – The QueryStrategy ranks the row numbers of the feature matrix. It returns an array of shape (n_instances,) containing the row indices in ranked order. If n_instances is None, returns all row numbers in ranked order. If n_instances is an integer, it only returns the top n_instances. If return_classifier_scores=True, also returns a second array with the same number of rows as the feature matrix, containing the relevance scores predicted by the classifier. If the classifier is not used, this will be None.

### asreview.models.query.UncertaintyQuery

**class** asreview.models.query.**UncertaintyQuery**

Uncertainty query strategy (uncertainty).

Choose the most uncertain samples according to the model (i.e. closest to 0.5 probability). Doesn't work very well in the case of LSTM's, since the probabilities are rather arbitrary.

### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

#### asreview.models.query.UncertaintyQuery.default_param

**property** UncertaintyQuery.**default_param**

Get the default parameters of the model.

**Returns**
*dict* – Dictionary with parameter: default value

#### asreview.models.query.UncertaintyQuery.label

UncertaintyQuery.**label** = **'Uncertainty'**

#### asreview.models.query.UncertaintyQuery.name

UncertaintyQuery.**name** = **'uncertainty'**

#### asreview.models.query.UncertaintyQuery.param

**property** UncertaintyQuery.**param**

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *query*(X, classifier[, n_instances, ...]) | Query method for strategies which use class probabilities. |

#### asreview.models.query.UncertaintyQuery.full_hyper_space

`UncertaintyQuery.`**`full_hyper_space`**`()`

#### asreview.models.query.UncertaintyQuery.hyper_space

`UncertaintyQuery.`**`hyper_space`**`()`

#### asreview.models.query.UncertaintyQuery.query

`UncertaintyQuery.`**`query`**`(`*X*, *classifier*, *n_instances=None*, *return_classifier_scores=False*, ***kwargs*`)`

Query method for strategies which use class probabilities.

### asreview.models.query.RandomQuery

**`class`** `asreview.models.query.`**`RandomQuery`**`(`*random_state=None*`)`

Random query strategy (`random`).

Randomly select samples with no regard to model assigned probabilities.

**Warning:** Selecting this option means your review is not going to be accelerated by ASReview.

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

#### asreview.models.query.RandomQuery.default_param

**`property`** `RandomQuery.`**`default_param`**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.query.RandomQuery.label

RandomQuery.**label** = **'Random'**

### asreview.models.query.RandomQuery.name

RandomQuery.**name** = **'random'**

### asreview.models.query.RandomQuery.param

**property** RandomQuery.**param**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *query*(X[, classifier, n_instances, ...]) | Put records in ranked order. |

### asreview.models.query.RandomQuery.full_hyper_space

RandomQuery.**full_hyper_space**()

### asreview.models.query.RandomQuery.hyper_space

RandomQuery.**hyper_space**()

### asreview.models.query.RandomQuery.query

RandomQuery.**query**(*X*, *classifier=None*, *n_instances=None*, *return_classifier_scores=False*, ***kwargs*)

Put records in ranked order.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix where every row contains the features of a record.
- **classifier** (*SKLearnModel*) – Trained classifier to compute relevance scores.
- **n_instances** (*int*) – Number of records to query. If None returns all records in ranked order.
- **return_classifier_score** (*bool*) – Return the relevance scores produced by the classifier.

**Returns**

*numpy.ndarray or (numpy.ndarray, np.ndarray)* – The QueryStrategy ranks the row numbers of the feature matrix. It returns an array of shape (n_instances,) containing the row indices in ranked order. If n_instances is None, returns all row numbers in ranked order. If n_instances is an integer, it only returns the top n_instances. If return_classifier_scores=True, also returns a second array with the same number of rows as the feature matrix, containing the relevance scores predicted by the classifier. If the classifier is not used, this will be None.

## asreview.models.query.ClusterQuery

**class** `asreview.models.query.`**`ClusterQuery`**(*cluster_size=350*, *update_interval=200*, *random_state=None*)

Clustering query strategy (`cluster`).

Use clustering after feature extraction on the dataset. Then the highest probabilities within random clusters are sampled.

**Parameters**

- **`cluster_size`** (*int*) – Size of the clusters to be made. If the size of the clusters is smaller than the size of the pool, fall back to max sampling.
- **`update_interval`** (*int*) – Update the clustering every x instances.
- **`random_state`** (*int, asreview.utils.SeededRandomState*) – State/seed of the RNG.

### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.query.ClusterQuery.default_param

**property** `ClusterQuery.`**`default_param`**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.query.ClusterQuery.label

`ClusterQuery.`**`label`** = **'Clustering'**

### asreview.models.query.ClusterQuery.name

`ClusterQuery.name = 'cluster'`

### asreview.models.query.ClusterQuery.param

`property ClusterQuery.param`

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *query*(X, classifier[, n_instances, ...]) | Query method for strategies which use class probabilities. |

### asreview.models.query.ClusterQuery.full_hyper_space

`ClusterQuery.full_hyper_space()`

### asreview.models.query.ClusterQuery.hyper_space

`ClusterQuery.hyper_space()`

### asreview.models.query.ClusterQuery.query

`ClusterQuery.query`(*X*, *classifier*, *n_instances=None*, *return_classifier_scores=False*, ***kwargs*)

Query method for strategies which use class probabilities.

Functions

| | |
|---|---|
| *query.get_query_model*(name, *args[, ...]) | Get an instance of the query strategy. |
| *query.get_query_class*(name) | Get class of query strategy from its name. |
| *query.list_query_strategies*() | List available query strategy classes. |

### asreview.models.query.get_query_model

`asreview.models.query.`**`get_query_model`**`(`*name*, **args*, *random_state=None*, ***kwargs*`)`

Get an instance of the query strategy.

**Parameters**

- **`name`** (*`str`*) – Name of the query strategy.
- **`*args`** – Arguments for the model.
- **`**kwargs`** – Keyword arguments for the model.

**Returns**

*asreview.query.base.BaseQueryModel* – Initialized instance of query strategy.

### asreview.models.query.get_query_class

`asreview.models.query.`**`get_query_class`**`(`*name*`)`

Get class of query strategy from its name.

**Parameters**

**`name`** (*`str`*) – Name of the query strategy, e.g. ‘max’, ‘uncertainty’, ‘random. A special mixed query strategy is als possible. The mix is denoted by an underscore: ‘max_random’ or ‘max_uncertainty’.

**Returns**

*class* – Class corresponding to the name name.

### asreview.models.query.list_query_strategies

`asreview.models.query.`**`list_query_strategies`**`()`

List available query strategy classes.

This excludes all possible mixed query strategies.

**Returns**

*list* – Classes of available query strategies in alphabetical order.

## 26.3.5 `asreview.models.balance`

Classes

| | |
|---|---|
| *`balance.base.BaseBalance`*() | Abstract class for balance strategies. |
| *`balance.SimpleBalance`*() | No balance strategy (`simple`). |
| *`balance.DoubleBalance`*([a, alpha, b, beta, ...]) | Double balance strategy (`double`). |
| *`balance.TripleBalance`*([a, alpha, b, beta, ...]) | Triple balance strategy (`triple`). |
| *`balance.UndersampleBalance`*([ratio, random_state]) | Undersampling balance strategy (`undersample`). |

### asreview.models.balance.base.BaseBalance

**class** asreview.models.balance.base.**BaseBalance**

Abstract class for balance strategies.

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

#### asreview.models.balance.base.BaseBalance.default_param

**property** BaseBalance.**default_param**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

#### asreview.models.balance.base.BaseBalance.name

BaseBalance.**name = 'base-balance'**

#### asreview.models.balance.base.BaseBalance.param

**property** BaseBalance.**param**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

#### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *sample*(X, y, train_idx) | Resample the training data. |

#### asreview.models.balance.base.BaseBalance.full_hyper_space

`BaseBalance.full_hyper_space()`

#### asreview.models.balance.base.BaseBalance.hyper_space

`BaseBalance.hyper_space()`

#### asreview.models.balance.base.BaseBalance.sample

`abstract BaseBalance.sample(X, y, train_idx)`

Resample the training data.

**Parameters**

- **X** (*numpy.ndarray*) – Complete feature matrix.
- **y** (*numpy.ndarray*) – Labels for all papers.
- **train_idx** (*numpy.ndarray*) – Training indices, that is all papers that have been reviewed.

**Returns**

*numpy.ndarray, numpy.ndarray* – X_train, y_train: the resampled matrix, labels.

### asreview.models.balance.SimpleBalance

`class asreview.models.balance.SimpleBalance`

No balance strategy (`simple`).

Use all training data.

#### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.balance.SimpleBalance.default_param

**property** SimpleBalance.**default_param**

Get the default parameters of the model.

**Returns**
*dict* – Dictionary with parameter: default value

### asreview.models.balance.SimpleBalance.label

SimpleBalance.**label = 'Simple (no balancing)'**

### asreview.models.balance.SimpleBalance.name

SimpleBalance.**name = 'simple'**

### asreview.models.balance.SimpleBalance.param

**property** SimpleBalance.**param**

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *sample*(X, y, train_idx) | Function that does not resample the training set. |

### asreview.models.balance.SimpleBalance.full_hyper_space

SimpleBalance.**full_hyper_space**()

### asreview.models.balance.SimpleBalance.hyper_space

SimpleBalance.**hyper_space**()

### asreview.models.balance.SimpleBalance.sample

`SimpleBalance.sample`(*X*, *y*, *train_idx*)

Function that does not resample the training set.

**Parameters**

- **X** (*numpy.ndarray*) – Complete matrix of all samples.
- **y** (*numpy.ndarray*) – Classified results of all samples.

**Returns**

- *numpy.ndarray* – Training samples.
- *numpy.ndarray* – Classification of training samples.

## asreview.models.balance.DoubleBalance

**class** `asreview.models.balance.DoubleBalance`(*a=2.155*, *alpha=0.94*, *b=0.789*, *beta=1.0*, *random_state=None*)

Double balance strategy (`double`).

Class to get the two way rebalancing function and arguments. It super samples ones depending on the number of 0's and total number of samples in the training data.

**Parameters**

- **a** (*float*) – Governs the weight of the 1's. Higher values mean linearly more 1's in your training sample.
- **alpha** (*float*) – Governs the scaling the weight of the 1's, as a function of the ratio of ones to zeros. A positive value means that the lower the ratio of zeros to ones, the higher the weight of the ones.
- **b** (*float*) – Governs how strongly we want to sample depending on the total number of samples. A value of 1 means no dependence on the total number of samples, while lower values mean increasingly stronger dependence on the number of samples.
- **beta** (*float*) – Governs the scaling of the weight of the zeros depending on the number of samples. Higher values means that larger samples are more strongly penalizing zeros.

### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.balance.DoubleBalance.default_param

**property** DoubleBalance.**default_param**

Get the default parameters of the model.

**Returns**
*dict* – Dictionary with parameter: default value

### asreview.models.balance.DoubleBalance.label

DoubleBalance.**label = 'Dynamic resampling (Double)'**

### asreview.models.balance.DoubleBalance.name

DoubleBalance.**name = 'double'**

### asreview.models.balance.DoubleBalance.param

**property** DoubleBalance.**param**

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *sample*(X, y, train_idx) | Resample the training data. |

### asreview.models.balance.DoubleBalance.full_hyper_space

DoubleBalance.**full_hyper_space**()

### asreview.models.balance.DoubleBalance.hyper_space

DoubleBalance.**hyper_space**()

### asreview.models.balance.DoubleBalance.sample

`DoubleBalance.sample`(*X*, *y*, *train_idx*)

Resample the training data.

**Parameters**

- **X** (*numpy.ndarray*) – Complete feature matrix.
- **y** (*numpy.ndarray*) – Labels for all papers.
- **train_idx** (*numpy.ndarray*) – Training indices, that is all papers that have been reviewed.

**Returns**

*numpy.ndarray,numpy.ndarray* – X_train, y_train: the resampled matrix, labels.

## asreview.models.balance.TripleBalance

`class asreview.models.balance.TripleBalance`(*a=2.155*, *alpha=0.94*, *b=0.789*, *beta=1.0*, *c=0.835*, *gamma=2.0*, *shuffle=True*, *random_state=None*)

Triple balance strategy (`triple`).

Broken. Only for internal and experimental use.

This divides the training data into three sets: included papers, excluded papers found with random sampling and papers found with max sampling. They are balanced according to formulas depending on the percentage of papers read in the dataset, the number of papers with random/max sampling etc. Works best for stochastic training algorithms. Reduces to both full sampling and undersampling with corresponding parameters.

**Parameters**

- **a** (*float*) – Governs the weight of the 1's. Higher values mean linearly more 1's in your training sample.
- **alpha** (*float*) – Governs the scaling the weight of the 1's, as a function of the ratio of ones to zeros. A positive value means that the lower the ratio of zeros to ones, the higher the weight of the ones.
- **b** (*float*) – Governs how strongly we want to sample depending on the total number of samples. A value of 1 means no dependence on the total number of samples, while lower values mean increasingly stronger dependence on the number of samples.
- **beta** (*float*) – Governs the scaling of the weight of the zeros depending on the number of samples. Higher values means that larger samples are more strongly penalizing zeros.
- **c** (*float*) – Value between one and zero that governs the weight of samples done with maximal sampling. Higher values mean higher weight.
- **gamma** (*float*) – Governs the scaling of the weight of the max samples as a function of the % of papers read. Higher values mean stronger scaling.

### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

#### asreview.models.balance.TripleBalance.default_param

**property** TripleBalance.**default_param**

Get the default parameters of the model.

**Returns**
*dict* – Dictionary with parameter: default value

#### asreview.models.balance.TripleBalance.label

TripleBalance.**label = 'Dynamic resampling (Triple)'**

#### asreview.models.balance.TripleBalance.name

TripleBalance.**name = 'triple'**

#### asreview.models.balance.TripleBalance.param

**property** TripleBalance.**param**

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *sample*(X, y, train_idx, shared) | Resample the training data. |

### asreview.models.balance.TripleBalance.full_hyper_space

`TripleBalance.full_hyper_space()`

### asreview.models.balance.TripleBalance.hyper_space

`TripleBalance.hyper_space()`

### asreview.models.balance.TripleBalance.sample

`TripleBalance.sample(X, y, train_idx, shared)`

Resample the training data.

**Parameters**

- **X** (*numpy.ndarray*) – Complete feature matrix.
- **y** (*numpy.ndarray*) – Labels for all papers.
- **train_idx** (*numpy.ndarray*) – Training indices, that is all papers that have been reviewed.
- **shared** (*dict*) – Dictionary to share data between balancing models and other models.

**Returns**

*numpy.ndarray,numpy.ndarray* – X_train, y_train: the resampled matrix, labels.

## asreview.models.balance.UndersampleBalance

`class asreview.models.balance.UndersampleBalance(ratio=1.0, random_state=None)`

Undersampling balance strategy (`undersample`).

This undersamples the data, leaving out excluded papers so that the included and excluded papers are in some particular ratio (closer to one).

**Parameters**

**ratio** (*double*) – Undersampling ratio of the zero's. If for example we set a ratio of 0.25, we would sample only a quarter of the zeros and all the ones.

### Attributes

| | |
|---|---|
| *default_param* | Get the default parameters of the model. |
| *label* | |
| *name* | |
| *param* | Get the (assigned) parameters of the model. |

### asreview.models.balance.UndersampleBalance.default_param

`property UndersampleBalance.default_param`

Get the default parameters of the model.

**Returns**
*dict* – Dictionary with parameter: default value

### asreview.models.balance.UndersampleBalance.label

`UndersampleBalance.label = 'Undersampling'`

### asreview.models.balance.UndersampleBalance.name

`UndersampleBalance.name = 'undersample'`

### asreview.models.balance.UndersampleBalance.param

`property UndersampleBalance.param`

Get the (assigned) parameters of the model.

**Returns**
*dict* – Dictionary with parameter: current value.

### Methods

| | |
|---|---|
| *full_hyper_space*() | |
| *hyper_space*() | |
| *sample*(X, y, train_idx) | Resample the training data. |

### asreview.models.balance.UndersampleBalance.full_hyper_space

`UndersampleBalance.full_hyper_space()`

### asreview.models.balance.UndersampleBalance.hyper_space

`UndersampleBalance.hyper_space()`

#### asreview.models.balance.UndersampleBalance.sample

`UndersampleBalance.`**`sample`**`(`*X*, *y*, *train_idx*`)`

Resample the training data.

**Parameters**

- **`X`** (*`numpy.ndarray`*) – Complete feature matrix.
- **`y`** (*`numpy.ndarray`*) – Labels for all papers.
- **`train_idx`** (*`numpy.ndarray`*) – Training indices, that is all papers that have been reviewed.

**Returns**

*numpy.ndarray,numpy.ndarray* – X_train, y_train: the resampled matrix, labels.

Functions

| | |
|---|---|
| *`balance.get_balance_model`*(name, *args[, ...]) | Get an instance of a balance model from a string. |
| *`balance.get_balance_class`*(name) | Get class of balance model from string. |
| *`balance.list_balance_strategies`*() | List available balancing strategy classes. |

### asreview.models.balance.get_balance_model

`asreview.models.balance.`**`get_balance_model`**`(`*name*, **args*, *random_state=None*, ***kwargs*`)`

Get an instance of a balance model from a string.

**Parameters**

- **`name`** (*`str`*) – Name of the balance model.
- **`*args`** – Arguments for the balance model.
- **`**kwargs`** – Keyword arguments for the balance model.

**Returns**

*BaseFeatureExtraction* – Initialized instance of features extraction algorithm.

### asreview.models.balance.get_balance_class

`asreview.models.balance.`**`get_balance_class`**`(`*name*`)`

Get class of balance model from string.

**Parameters**

**`name`** (*`str`*) – Name of the model, e.g. ‘simple’, ‘double’ or ‘undersample’.

**Returns**

*BaseBalanceModel* – Class corresponding to the name.

#### asreview.models.balance.list_balance_strategies

`asreview.models.balance.list_balance_strategies()`

List available balancing strategy classes.

**Returns**

*list* – Classes of available balance strategies in alphabetical order.

## 26.4 Projects and States

Load, interact, and extract information from project files and states (the "diary" of the review).

### 26.4.1 Project

| | |
|---|---|
| *Project*(project_path[, project_id]) | Project class for ASReview project files. |

#### asreview.Project

`class asreview.Project(`*project_path*, *project_id=None*`)`

Project class for ASReview project files.

##### Attributes

| |
|---|
| *config* |
| *feature_matrices* |
| *reviews* |

#### asreview.Project.config

`property Project.config`

### asreview.Project.feature_matrices

**property** Project.**feature_matrices**

### asreview.Project.reviews

**property** Project.**reviews**

### Methods

| | |
|---|---|
| *add_dataset*(file_name) | Add file path to the project file. |
| *add_feature_matrix*(feature_matrix, ...) | Add feature matrix to project file. |
| *add_review*(review_id[, start_time, status]) | Add new review metadata. |
| *clean_tmp_files*() | Clean temporary files in a project. |
| *create*(project_path[, project_id, ...]) | Initialize the necessary files specific to the web app. |
| *delete_review*([remove_folders]) | |
| *export*(export_fp) | |
| *get_feature_matrix*(feature_extraction_method) | Get the feature matrix from the project file. |
| *load*(asreview_file, project_path[, safe_import]) | |
| *mark_review_finished*([review_id]) | Mark a review in the project as finished. |
| *read_data*([use_cache, save_cache]) | Get Dataset object from file. |
| *remove_dataset*() | Remove dataset from project. |
| *remove_error*(status) | |
| *set_error*(err[, save_error_message]) | |
| *update_config*(**kwargs) | Update project info |
| *update_review*([review_id]) | Update review metadata. |

### asreview.Project.add_dataset

Project.**add_dataset**(*file_name*)

Add file path to the project file.

Add file to data subfolder and fill the pool of iteration 0.

### asreview.Project.add_feature_matrix

`Project.`**`add_feature_matrix`**(*feature_matrix*, *feature_extraction_method*)

Add feature matrix to project file.

**Parameters**

- **`feature_matrix`** (*`numpy.ndarray`*, *`scipy.sparse.csr.csr_matrix`*) – The feature matrix to add to the project file.
- **`feature_extraction_method`** (*`str`*) – Name of the feature extraction method.

### asreview.Project.add_review

`Project.`**`add_review`**(*review_id*, *start_time=None*, *status='setup'*)

Add new review metadata.

**Parameters**

- **`review_id`** (*`str`*) – The review_id uuid4.
- **`status`** (*`str`*) – The status of the review. One of 'setup', 'running', 'finished'.
- **`start_time`** – Start of the review.

### asreview.Project.clean_tmp_files

`Project.`**`clean_tmp_files`**()

Clean temporary files in a project.

**Parameters**

**`project_id`** (*`str`*) – The id of the current project.

### asreview.Project.create

**`classmethod`** `Project.`**`create`**(*project_path*, *project_id=None*, *project_mode='oracle'*, *project_name=None*, *project_description=None*, *project_authors=None*, *project_tags=None*)

Initialize the necessary files specific to the web app.

### asreview.Project.delete_review

`Project.`**`delete_review`**(*remove_folders=False*)

### asreview.Project.export

`Project.`**`export`**(*export_fp*)

### asreview.Project.get_feature_matrix

`Project.`**`get_feature_matrix`**(*feature_extraction_method*)

Get the feature matrix from the project file.

**Parameters**

**`feature_extraction_method`** (*str*) – Name of the feature extraction method for which to get the matrix.

**Returns**

*scipy.sparse.csr_matrix* – Feature matrix in sparse format.

### asreview.Project.load

`classmethod Project.`**`load`**(*asreview_file*, *project_path*, *safe_import=False*)

### asreview.Project.mark_review_finished

`Project.`**`mark_review_finished`**(*review_id=None*)

Mark a review in the project as finished.

If no review_id is given, mark the first review as finished.

**Parameters**

**`review_id`** (*str*) – Identifier of the review to mark as finished.

### asreview.Project.read_data

`Project.`**`read_data`**(*use_cache=True*, *save_cache=True*)

Get Dataset object from file.

**Parameters**

- **`use_cache`** (*bool*) – Use the pickle file if available.
- **`save_cache`** (*bool*) – Save the file to a pickle file if not available.

**Returns**

*Dataset* – The data object for internal use in ASReview.

#### asreview.Project.remove_dataset

`Project.`**`remove_dataset`**`()`

Remove dataset from project.

#### asreview.Project.remove_error

`Project.`**`remove_error`**`(`*status*`)`

#### asreview.Project.set_error

`Project.`**`set_error`**`(`*err*, *save_error_message=True*`)`

#### asreview.Project.update_config

`Project.`**`update_config`**`(`***kwargs*`)`

Update project info

#### asreview.Project.update_review

`Project.`**`update_review`**`(`*review_id=None*, ***kwargs*`)`

Update review metadata.

**Parameters**

- **`review_id`** (*`str`*) – The review_id uuid4. Default None, which is the first added review.
- **`status`** (*`str`*) – The status of the review. One of 'setup', 'running', 'finished'.
- **`start_time`** – Start of the review.
- **`end_time`** (*`End time of the review.`*) –

### 26.4.2 State

| | |
|---|---|
| *`open_state`*(asreview_obj[, review_id, read_only]) | Initialize a state class instance from a project folder. |
| *`state.SQLiteState`*([read_only]) | Class for storing the review state. |

#### asreview.open_state

`asreview.`**`open_state`**`(`*asreview_obj*, *review_id=None*, *read_only=True*`)`

Initialize a state class instance from a project folder.

**Parameters**

- **`asreview_obj`** (*`str/pathlike/Project`*) – Filepath to the (unzipped) project folder or Project object.

- **review_id** (*str*) – Identifier of the review from which the state will be instantiated. If none is given, the first review in the reviews folder will be taken.
- **read_only** (*bool*) – Whether to open in read_only mode.

**Returns**
*SQLiteState*

## asreview.state.SQLiteState

**class** asreview.state.**SQLiteState**(*read_only=True*)

Class for storing the review state.

The results are stored in a sqlite database.

**Parameters**
**read_only** (*bool*) – Open state in read only mode. Default False.

**Variables**

- **version** (*str*) – Return the version number of the state.
- **settings** (asreview.settings.ASReviewSettings) – Return an ASReview settings object with model settings and active learning settings.
- **n_records_labeled** (*int*) – Get the number of labeled records, where each prior is counted individually.
- **n_priors** (*int*) – Number of priors. If priors have not been selected returns None.
- **exist_new_labeled_records** (*bool*) – Have there been labeled records added to the state since the last time a model ranking was added to the state?
- **model_has_trained** (*bool*) – Has the ranking by a model been added to the state?

### Attributes

| | |
|---|---|
| *exist_new_labeled_records* | Return True if there are new labeled records. |
| *model_has_trained* | Return True if there is data of a trained model in the state. |
| *n_priors* | Number of records added as prior knowledge. |
| *n_records* | Number of records in the loop. |
| *n_records_labeled* | Number labeled records. |
| *settings* | Settings of the ASReview pipeline. |
| *version* | Version number of the state. |

### asreview.state.SQLiteState.exist_new_labeled_records

**property** `SQLiteState.`**`exist_new_labeled_records`**

Return True if there are new labeled records.

Return True if there are any record labels added since the last time the model ranking was added to the state. Also returns True if no model was trained yet, but priors have been added.

### asreview.state.SQLiteState.model_has_trained

**property** `SQLiteState.`**`model_has_trained`**

Return True if there is data of a trained model in the state.

### asreview.state.SQLiteState.n_priors

**property** `SQLiteState.`**`n_priors`**

Number of records added as prior knowledge.

**Returns**

*int* – Number of records which were added as prior knowledge.

### asreview.state.SQLiteState.n_records

**property** `SQLiteState.`**`n_records`**

Number of records in the loop.

**Returns**

*int* – Number of records.

### asreview.state.SQLiteState.n_records_labeled

**property** `SQLiteState.`**`n_records_labeled`**

Number labeled records.

**Returns**

*int* – Number of labeled records, priors counted individually.

### asreview.state.SQLiteState.settings

**property** `SQLiteState.`**`settings`**

Settings of the ASReview pipeline.

### Example

Example of settings.

model : nb query_strategy : max_random balance_strategy : triple feature_extraction: tfidf n_instances : 1 stop_if : min n_prior_included : 10 n_prior_excluded : 10 mode : simulate model_param : {'alpha': 3.822} query_param : {'strategy_1': 'max', 'strategy_2': 'random', 'mix_ratio': 0.95} feature_param : {} balance_param : {'a': 2.155, 'alpha': 0.94, … 'gamma': 2.0, 'shuffle': True} abstract_only : False

## asreview.state.SQLiteState.version

**property** `SQLiteState.version`

Version number of the state.

**Returns**

*str* – Returns the version of the state.

## Methods

| | |
|---|---|
| *add_labeling_data*(record_ids, labels[, ...]) | Add the data corresponding to a labeling action to the state file. |
| *add_last_probabilities*(probabilities) | Save the probabilities produced by the last classifier. |
| *add_last_ranking*(ranked_record_ids, ...) | Save the ranking of the last iteration of the model. |
| *add_note*(note, record_id) | Add a text note to save with a labeled record. |
| *add_record_table*(record_ids) | Add the record table to the state. |
| *close*() | Close the files opened by the state. |
| *connect_to_sql_r*() | Get a connection to the SQLite database. |
| *connect_to_sql_wr*() | Get a connection to the SQLite database. |
| *delete_record_labeling_data*(record_id) | Delete the labeling data for the given record id. |
| *get_balance_strategies*([priors, pending]) | Get the balance strategies from the state. |
| *get_classifiers*([priors, pending]) | Get the classifiers from the state. |
| *get_data_by_query_number*(query[, columns]) | Get the data of a specific query from the results table. |
| *get_data_by_record_id*(record_id[, columns]) | Get the data of a specific query from the results table. |
| *get_dataset*([columns, priors, pending]) | Get a subset from the results table. |
| *get_decision_changes*() | Get the record ids for any decision changes. |
| *get_feature_extraction*([priors, pending]) | Get the query strategies from the state. |
| *get_labeled*() | Get the labeled records in order of labeling. |
| *get_labeling_times*([time_format, priors, ...]) | Get the time of labeling from the state. |
| *get_labels*([priors, pending]) | Get the labels from the state. |
| *get_last_probabilities*() | Get the probabilities produced by the last classifier. |
| *get_last_ranking*() | Get the ranking from the state. |
| *get_order_of_labeling*([priors, pending]) | Get full array of record id's in order that they were labeled. |
| *get_pending*() | Get the record_ids of the records pending a labeling decision. |
| *get_pool*() | Get the unlabeled, not-pending records in ranking order. |
| *get_pool_labeled_pending*() | Return the unlabeled pool, labeled and pending records. |
| *get_priors*([columns]) | Get the record ids of the priors. |

continues on next page

Table 1 – continued from previous page

| | |
|---|---|
| *get_query_strategies*([priors, pending]) | Get the query strategies from the state. |
| *get_record_table*() | Get the record table of the state. |
| *get_training_sets*([priors, pending]) | Get the training_sets from the state. |
| *is_empty*() | Check if state has no results. |
| *query_top_ranked*(n) | Get the top ranked records from the ranking table. |
| *to_dict*() | Convert state to dictionary. |
| *update_decision*(record_id, label[, note, tags]) | Change the label of an already labeled record. |

### asreview.state.SQLiteState.add_labeling_data

SQLiteState.**add_labeling_data**(*record_ids*, *labels*, *notes=None*, *tags_list=None*, *prior=False*)

Add the data corresponding to a labeling action to the state file.

**Parameters**

- **record_ids** (*list*, *numpy.ndarray*) – A list of ids of the labeled records as int.
- **labels** (*list*, *numpy.ndarray*) – A list of labels of the labeled records as int.
- **notes** (*list of str/None*) – A list of text notes to save with the labeled records.
- **tags_list** (*list of list*) – A list of tags to save with the labeled records.
- **prior** (*bool*) – Whether the added record are prior knowledge.

### asreview.state.SQLiteState.add_last_probabilities

SQLiteState.**add_last_probabilities**(*probabilities*)

Save the probabilities produced by the last classifier.

**Parameters**

**probabilities** (*list*, *np.array*) – List containing the relevance scores for every record. If this is None, the last probabilities table in the state is emptied.

### asreview.state.SQLiteState.add_last_ranking

SQLiteState.**add_last_ranking**(*ranked_record_ids*, *classifier*, *query_strategy*, *balance_strategy*, *feature_extraction*, *training_set*)

Save the ranking of the last iteration of the model.

Save the ranking of the last iteration of the model, in the ranking order, so the record on row 0 is ranked first by the model.

**Parameters**

- **ranked_record_ids** (*list*, *numpy.ndarray*) – A list of records ids in the order that they were ranked.
- **classifier** (*str*) – Name of the classifier of the model.
- **query_strategy** (*str*) – Name of the query strategy of the model.
- **balance_strategy** (*str*) – Name of the balance strategy of the model.
- **feature_extraction** (*str*) – Name of the feature extraction method of the model.

- **training_set** (*int*) – Number of labeled records available at the time of training.

### asreview.state.SQLiteState.add_note

SQLiteState.**add_note**(*note*, *record_id*)

Add a text note to save with a labeled record.

**Parameters**

- **note** (*str*) – Text note to save.
- **record_id** (*int*) – Identifier of the record to which the note should be added.

### asreview.state.SQLiteState.add_record_table

SQLiteState.**add_record_table**(*record_ids*)

Add the record table to the state.

**Parameters**

**record_ids** (*list, np.array*) – List containing all record ids of the dataset.

### asreview.state.SQLiteState.close

SQLiteState.**close**()

Close the files opened by the state.

### asreview.state.SQLiteState.connect_to_sql_r

SQLiteState.**connect_to_sql_r**()

Get a connection to the SQLite database.

**Returns**

*sqlite3.Connection* – Read only connection to the SQLite database.

### asreview.state.SQLiteState.connect_to_sql_wr

SQLiteState.**connect_to_sql_wr**()

Get a connection to the SQLite database.

**Returns**

*sqlite3.Connection* – Write / read connection to the SQLite database.

### asreview.state.SQLiteState.delete_record_labeling_data

`SQLiteState.`**`delete_record_labeling_data`**`(`*record_id*`)`

Delete the labeling data for the given record id.

**Parameters**

**`record_id`** (*`str`*) – Identifier of the record to delete.

### asreview.state.SQLiteState.get_balance_strategies

`SQLiteState.`**`get_balance_strategies`**`(`*priors=True*, *pending=False*`)`

Get the balance strategies from the state.

**Parameters**

- **`priors`** (*`bool`*) – Whether to keep the records containing the prior knowledge.
- **`pending`** (*`bool`*) – Whether to keep the records which are pending a labeling decision.

**Returns**

*pd.Series* – Series containing the balance strategy used to get the training data at each labeling moment.

### asreview.state.SQLiteState.get_classifiers

`SQLiteState.`**`get_classifiers`**`(`*priors=True*, *pending=False*`)`

Get the classifiers from the state.

**Parameters**

- **`priors`** (*`bool`*) – Whether to keep the records containing the prior knowledge.
- **`pending`** (*`bool`*) – Whether to keep the records which are pending a labeling decision.

**Returns**

*pd.Series* – Series containing the classifier used at each labeling moment.

### asreview.state.SQLiteState.get_data_by_query_number

`SQLiteState.`**`get_data_by_query_number`**`(`*query*, *columns=None*`)`

Get the data of a specific query from the results table.

**Parameters**

- **`query`** (*`int`*) – Number of the query of which you want the data. query=0 corresponds to all the prior records.
- **`columns`** (*`list`*) – List of columns names of the results table.

**Returns**

*pd.DataFrame* – Dataframe containing the data from the results table with the given query number and columns.

### asreview.state.SQLiteState.get_data_by_record_id

`SQLiteState.get_data_by_record_id(record_id, columns=None)`

Get the data of a specific query from the results table.

**Parameters**

- **record_id** (*int*) – Record id of which you want the data.
- **columns** (*list*) – List of columns names of the results table.

**Returns**

*pd.DataFrame* – Dataframe containing the data from the results table with the given record_id and columns.

### asreview.state.SQLiteState.get_dataset

`SQLiteState.get_dataset(columns=None, priors=True, pending=False)`

Get a subset from the results table.

Can be used to get any column subset from the results table. Most other get functions use this one, except some that use a direct SQL query for efficiency.

**Parameters**

- **columns** (*list, str*) – List of columns names of the results table, or a string containing one column name.
- **priors** (*bool*) – Whether to keep the records containing the prior knowledge.
- **pending** (*bool*) – Whether to keep the records which are pending a labeling decision.

**Returns**

*pd.DataFrame* – Dataframe containing the data of the specified columns of the results table.

### asreview.state.SQLiteState.get_decision_changes

`SQLiteState.get_decision_changes()`

Get the record ids for any decision changes.

Get the record ids of the records whose labels have been changed after the original labeling action.

**Returns**

*pd.DataFrame* – Dataframe with columns 'record_id', 'new_label', and 'time' for each record of which the labeling decision was changed.

### asreview.state.SQLiteState.get_feature_extraction

`SQLiteState.get_feature_extraction(priors=True, pending=False)`

Get the query strategies from the state.

**Parameters**

- **priors** (*bool*) – Whether to keep the records containing the prior knowledge.
- **pending** (*bool*) – Whether to keep the records which are pending a labeling decision.

**Returns**
*pd.Series* – Series containing the feature extraction method used for the classifier input at each labeling moment.

### asreview.state.SQLiteState.get_labeled

SQLiteState.**get_labeled**()

Get the labeled records in order of labeling.

Get the record_ids and labels of the labeled records in order of labeling. If you only want the labeled records, this is more efficient than via 'get_pool_labeled_pending'.

**Returns**
*pd.DataFrame* – Dataframe containing the record_ids and labels of the labeled records, in the order that they were labeled.

### asreview.state.SQLiteState.get_labeling_times

SQLiteState.**get_labeling_times**(*time_format='int'*, *priors=True*, *pending=False*)

Get the time of labeling from the state.

**Parameters**

- **time_format** (*'int' or 'datetime'*) – Format of the return value. If it is 'int' you get a UTC timestamp, if it is 'datetime' you get datetime instead of an integer.
- **priors** (*bool*) – Whether to keep the records containing the prior knowledge.
- **pending** (*bool*) – Whether to keep the records which are pending a labeling decision.

**Returns**
*pd.Series* – If format='int' you get a UTC timestamp (integer number of microseconds), if it is 'datetime' you get datetime format.

### asreview.state.SQLiteState.get_labels

SQLiteState.**get_labels**(*priors=True*, *pending=False*)

Get the labels from the state.

**Parameters**

- **priors** (*bool*) – Whether to keep the records containing the prior knowledge.
- **pending** (*bool*) – Whether to keep the records which are pending a labeling decision.

**Returns**
*pd.Series* – Series containing the labels at each labelling moment.

### asreview.state.SQLiteState.get_last_probabilities

`SQLiteState.`**`get_last_probabilities`**`()`

Get the probabilities produced by the last classifier.

**Returns**

*pd.Series* – Series with name 'proba' containing the probabilities.

### asreview.state.SQLiteState.get_last_ranking

`SQLiteState.`**`get_last_ranking`**`()`

Get the ranking from the state.

**Returns**

*pd.DataFrame* – Dataframe with columns 'record_id', 'ranking', 'classifier', 'query_strategy', 'balance_strategy', 'feature_extraction', 'training_set' and 'time'. It has one row for each record in the dataset, and is ordered by ranking.

### asreview.state.SQLiteState.get_order_of_labeling

`SQLiteState.`**`get_order_of_labeling`**`(`*priors=True*, *pending=False*`)`

Get full array of record id's in order that they were labeled.

**Parameters**

- **`priors`** (*`bool`*) – Whether to keep the records containing the prior knowledge.
- **`pending`** (*`bool`*) – Whether to keep the records are pending a labeling decision.

**Returns**

*pd.Series* – The record_id's in the order that they were labeled.

### asreview.state.SQLiteState.get_pending

`SQLiteState.`**`get_pending`**`()`

Get the record_ids of the records pending a labeling decision.

If you only want the pending records, this is more efficient than via 'get_pool_labeled_pending'.

**Returns**

*pd.Series* – A series containing the record_ids of the records whose label is pending.

### asreview.state.SQLiteState.get_pool

`SQLiteState.`**`get_pool`**`()`

Get the unlabeled, not-pending records in ranking order.

Get the pool of unlabeled records, not pending a labeling decision, in the ranking order. If you only want the records in the pool, this is more efficient than via 'get_pool_labeled_pending'.

**Returns**

*pd.Series* – Series containing the record_ids of the unlabeled, not pending records, in the order of the last available ranking.

### asreview.state.SQLiteState.get_pool_labeled_pending

`SQLiteState.get_pool_labeled_pending()`

Return the unlabeled pool, labeled and pending records.

Convenience function to get the pool, labeled and pending records in one SQL query. If you only want one of these, it is more efficient to use the methods 'get_pool', 'get_labeled' or 'get_pending'.

**Returns**

*tuple (pd.Series, pd.DataFrame, pd.Series)* – Returns a tuple (pool, labeled, pending). Pool is a series containing the unlabeled, not pending record_ids, ordered by the last predicted ranking of the model. Labeled is a dataframe containing the record_ids and labels of the labeled records, in the order that they were labeled. Pending is a series containing the record_ids of the records whose label is pending.

### asreview.state.SQLiteState.get_priors

`SQLiteState.get_priors(`*columns=None*`)`

Get the record ids of the priors.

**Returns**

*pd.Series* – The record_id's of the priors in the order they were added.

### asreview.state.SQLiteState.get_query_strategies

`SQLiteState.get_query_strategies(`*priors=True*, *pending=False*`)`

Get the query strategies from the state.

**Parameters**

- **priors** (*bool*) – Whether to keep the records containing the prior knowledge.
- **pending** (*bool*) – Whether to keep the records which are pending a labeling decision.

**Returns**

*pd.Series* – Series containing the query strategy used to get the record to query at each labeling moment.

### asreview.state.SQLiteState.get_record_table

`SQLiteState.get_record_table()`

Get the record table of the state.

**Returns**

*pd.Series* – Series with name 'record_id' containing the record ids.

### asreview.state.SQLiteState.get_training_sets

`SQLiteState.`**`get_training_sets`**`(`*priors=True*, *pending=False*`)`

Get the training_sets from the state.

**Parameters**

- **`priors`** (*`bool`*) – Whether to keep the records containing the prior knowledge.
- **`pending`** (*`bool`*) – Whether to keep the records which are pending a labeling decision.

**Returns**

*pd.Series* – Series containing the training set on which the classifier was fit at each labeling moment.

### asreview.state.SQLiteState.is_empty

`SQLiteState.`**`is_empty`**`()`

Check if state has no results.

**Returns**

*bool* – True if empty.

### asreview.state.SQLiteState.query_top_ranked

`SQLiteState.`**`query_top_ranked`**`(`*n*`)`

Get the top ranked records from the ranking table.

Get the top n instances from the pool according to the last ranking. Add the model data to the results table.

**Parameters**

**`n`** (*`int`*) – Number of instances.

**Returns**

*list* – List of record_ids of the top n ranked records.

### asreview.state.SQLiteState.to_dict

`SQLiteState.`**`to_dict`**`()`

Convert state to dictionary.

**Returns**

*dict* – Dictionary with all settings and results.

#### asreview.state.SQLiteState.update_decision

`SQLiteState.`**`update_decision`**(*record_id*, *label*, *note=None*, *tags=None*)

Change the label of an already labeled record.

**Parameters**

- **record_id** (*int*) – Id of the record whose label should be changed.
- **label** (*0 / 1*) – New label of the record.
- **note** (*str*) – Note to add to the record.
- **tags** (*list*) – Tags list to add to the record.

### 26.4.3 Utils

| | |
|---|---|
| *project.get_project_path*(folder_id) | Get the project directory. |
| *project.project_from_id*(f) | Decorator function that takes a user account as parameter, the user account is used to get the correct sub folder in which the projects is |
| *project.get_projects*([project_paths]) | Get the ASReview projects at the given paths. |
| *project.is_project*(project_path) | |
| *project.is_v0_project*(project_path) | Check if a project file is of a ASReview version 0 project. |

#### asreview.project.get_project_path

`asreview.project.`**`get_project_path`**(*folder_id*)

Get the project directory.

**Parameters**

**folder_id** (*str*) – The id of the folder containing a project. If there is no authentication, the folder_id is equal to the project_id. Otherwise, this is equal to {project_owner_id}_{project_id}.

#### asreview.project.project_from_id

`asreview.project.`**`project_from_id`**(*f*)

Decorator function that takes a user account as parameter, the user account is used to get the correct sub folder in which the projects is

#### asreview.project.get_projects

`asreview.project.`**`get_projects`**(*project_paths=None*)

Get the ASReview projects at the given paths.

**Parameters**

**project_paths** (*list[Path], optional*) – List of paths to projects. By default all the projects in the asreview folder are used, by default None

**Returns**

*list[Project]* – Projects at the given project paths.

#### asreview.project.is_project

asreview.project.**is_project**(*project_path*)

#### asreview.project.is_v0_project

asreview.project.**is_v0_project**(*project_path*)

Check if a project file is of a ASReview version 0 project.

## 26.5 Misc

Classes

| | |
|---|---|
| *asreview.settings.ASReviewSettings*(model, ...) | Object to store the configuration of a review session. |

### 26.5.1 asreview.settings.ASReviewSettings

**class** asreview.settings.**ASReviewSettings**(*model*, *query_strategy*, *balance_strategy*, *feature_extraction*, *n_instances=1*, *stop_if=None*, *n_prior_included=None*, *n_prior_excluded=None*, *as_data=None*, *model_param=None*, *query_param=None*, *balance_param=None*, *feature_param=None*, ***kwargs*)

Object to store the configuration of a review session.

The main difference being that it type checks (some) of its contents.

#### Methods

| | |
|---|---|
| *from_file*(config_file) | Fill the contents of settings by reading a config file. |
| *to_dict*() | Export default settings to dict. |

#### asreview.settings.ASReviewSettings.from_file

ASReviewSettings.**from_file**(*config_file*)

Fill the contents of settings by reading a config file.

**Parameters**

**config_file** (*str*) – Source configuration file.

#### asreview.settings.ASReviewSettings.to_dict

`ASReviewSettings.to_dict()`

Export default settings to dict.

Functions

| | |
|---|---|
| *search.fuzzy_find*(as_data, keywords[, ...]) | Find a record using keywords. |
| *asreview_path*() | Get the location where projects are stored. |
| *get_data_home*([data_home]) | Return the path of the ASR data dir. |

### 26.5.2 asreview.search.fuzzy_find

`asreview.search.fuzzy_find`(*as_data*, *keywords*, *threshold=60*, *max_return=10*, *exclude=None*)

Find a record using keywords.

It looks for keywords in the title/authors/keywords (for as much is available). Using the diflib package it creates a ranking based on token set matching.

**Parameters**

- **as_data** (`asreview.Dataset`) – ASReview data object to search
- **keywords** (*str*) – A string of keywords together, can be a combination.
- **threshold** (*float*) – Don't return records below this threshold.
- **max_return** (*int*) – Maximum number of records to return.
- **exclude** (*list, numpy.ndarray*) – List of indices that should be excluded in the search. You would put papers that were already labeled here for example.

**Returns**

*list* – Sorted list of indexes that match best the keywords.

### 26.5.3 asreview.asreview_path

`asreview.asreview_path()`

Get the location where projects are stored.

Overwrite this location by specifying the ASREVIEW_PATH enviroment variable.

### 26.5.4 asreview.get_data_home

`asreview.get_data_home`(*data_home=None*)

Return the path of the ASR data dir.

This folder is used by some large dataset loaders to avoid downloading the data several times. By default the data dir is set to a folder named 'asr_data' in the user home folder. Alternatively, it can be set by the 'ASR_DATA' environment variable or programmatically by giving an explicit folder path. The '~' symbol is expanded to the user home folder. If the folder does not already exist, it is automatically created.

**Parameters**

**data_home** (*str | None*) – The path to scikit-learn data dir.

# CHAPTER TWENTYSEVEN

# INDICES AND TABLES

- genindex
- modindex

# PYTHON MODULE INDEX

# INDEX