

---

# **ASReview Software Documentation**

***Release 1.3***

**ASReview LAB developers, Utrecht University**

**Nov 02, 2023**



## CONTENTS

<b>1</b>	<b>Get Started</b>	<b>3</b>
<b>2</b>	<b>Research</b>	<b>9</b>
<b>3</b>	<b>Support</b>	<b>11</b>
<b>4</b>	<b>Installation</b>	<b>13</b>
<b>5</b>	<b>Start ASReview LAB</b>	<b>17</b>
<b>6</b>	<b>Troubleshooting</b>	<b>21</b>
<b>7</b>	<b>Prepare your data</b>	<b>23</b>
<b>8</b>	<b>Data format</b>	<b>27</b>
<b>9</b>	<b>Fully and partially labeled data</b>	<b>31</b>
<b>10</b>	<b>Create a project</b>	<b>33</b>
<b>11</b>	<b>Screening</b>	<b>43</b>
<b>12</b>	<b>Progress and results</b>	<b>49</b>
<b>13</b>	<b>Manage projects</b>	<b>55</b>
<b>14</b>	<b>Extensions</b>	<b>57</b>
<b>15</b>	<b>Overview</b>	<b>59</b>
<b>16</b>	<b>Simulate via the webapp</b>	<b>61</b>
<b>17</b>	<b>Simulation via command line</b>	<b>63</b>
<b>18</b>	<b>Analyzing results</b>	<b>67</b>
<b>19</b>	<b>Simulate with Python API</b>	<b>69</b>
<b>20</b>	<b>Overview</b>	<b>73</b>
<b>21</b>	<b>Command Line</b>	<b>75</b>
<b>22</b>	<b>Access data from ASReview file</b>	<b>77</b>

<b>23 Create extensions</b>	<b>85</b>
<b>24 API Reference</b>	<b>89</b>
<b>25 Indices and tables</b>	<b>199</b>
<b>Python Module Index</b>	<b>201</b>
<b>Index</b>	<b>203</b>

Welcome to the ASReview LAB Documentation!



## 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.
- **Exploration:** Explore or demonstrate ASReview LAB with a completely labeled dataset. This mode is suitable for teaching purposes.

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, Exploration, 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 Exploration 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 exploration in ASReview LAB:

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

**Exploration** mode is used when a user explores or demonstrates ASReview LAB with a completely labeled dataset. This mode is suitable for teaching purposes.

**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 exploration mode, the user adds labels to *records*, or in simulation mode, the simulation is running.

**Finished** refers to the fact that in oracle or exploration 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!



## 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!

### 2.1 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.

## 2.2 Cite

For scientific use, we encourage users to cite:

- The paper published in [Nature Machine Intelligence](#) to cite the **ASReview project**.
- For citing the software **ASReview LAB**, refer to the [specific release](#) of the software. The menu on the right (in Zenodo) can be used to find the citation format of prevalence.
- For citing the documentation (or to download the pdf) go to [Zenodo](#).
- More studies related to the project can be found on [asreview.ai/research](https://asreview.ai/research).

## SUPPORT

Questions can be asked on [GitHub Discussions](#). For bug reports and feature requests, please submit an issue on [GitHub](#).

### 3.1 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!

### 3.2 Collaborate

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

### 3.3 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](#).





## INSTALLATION

### 4.1 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](#).

---

### 4.2 Upgrade ASReview

Upgrade ASReview software with

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

## 4.3 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.

## 4.4 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](#).

## 4.5 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.

### 4.5.1 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
```

### 4.5.2 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`.

### 4.5.3 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
```



## 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.

### 5.1 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 hashing 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  $2^{32} - 1$ .

## 5.2 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%
```

## 5.3 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
```





## TROUBLESHOOTING

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

### 6.1 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
```

### 6.2 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
```

## 6.3 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.

## 6.4 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.

## 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*. The latter ones are useful in the Simulation and Exploration mode. 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*.

### 7.1 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.

### 7.2 Compatibility

#### 7.2.1 Citation Managers

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

	<b>.ris</b>	<b>.csv</b>	<b>.xlsx</b>
<b>EndNote</b>		N/A	N/A
<b>Excel</b>	N/A		
<b>Mendeley</b>		N/A	N/A
<b>Refworks</b>		N/A	N/A
<b>Zotero</b>			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”.
- 

## 7.2.2 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
<b>CINAHL (EBSCO)</b>		N/A	X	N/A
<b>Cochrane</b>		N/A		N/A
<b>Embase</b>		N/A		
<b>Eric (Ovid)</b>	*	N/A	N/A	N/A
<b>Psychinfo (Ovid)</b>	*	N/A	N/A	N/A
<b>Pubmed</b>	X	N/A	X	N/A
<b>Scopus</b>		N/A		N/A
<b>Web of Science</b>		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.

---

### 7.2.3 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.

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



## 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:

### 8.1 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. Similarly, 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, and 1 = relevant/included. Different column names are allowed, see the table. It can be used for:

- **Screening:** In ASReview LAB, if labels are available for a part of the dataset (see *Fully and partially labeled data*), 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.
- **Exploration:** You can explore a completely labeled dataset in the Exploration Mode. The relevant/irrelevant label in the dataset will be displayed on each record. This option is useful for training purposes, presentations, and workshops.
- **Simulation:** In a *Simulation*, the column containing the labels is used to simulate a systematic review run. Only records containing labels are used for the simulation, unlabeled records are ignored.

---

**Note:** Files exported with ASReview LAB contain the column `included`. When re-importing a partly labeled dataset in 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.

---

## 8.2 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.

---

**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 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.

---



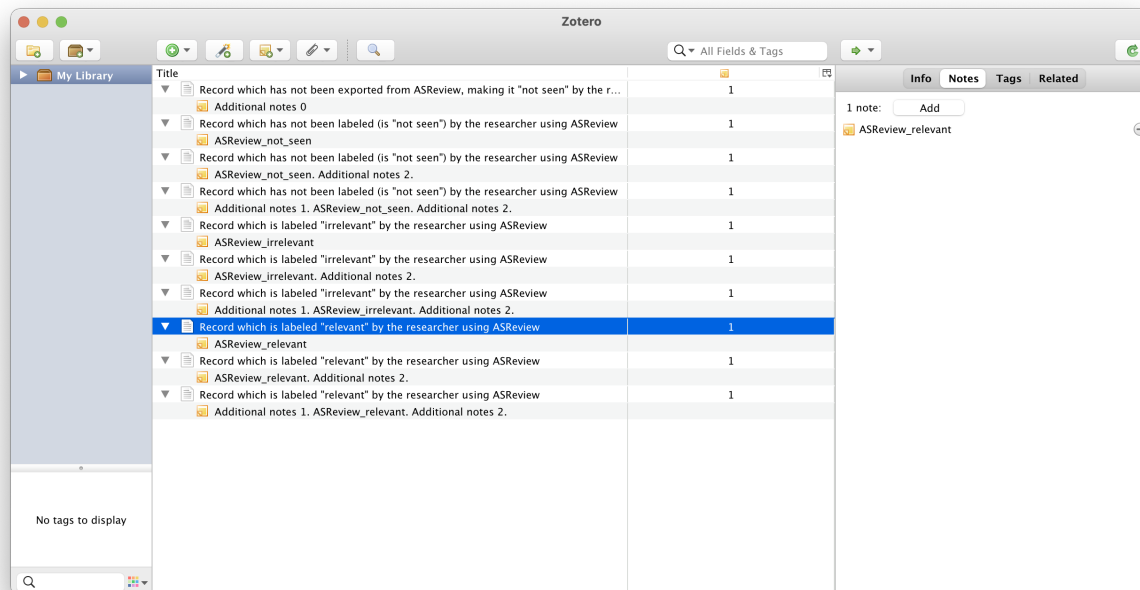


Fig. 1: Example record with a labeling decision imported to Zotero

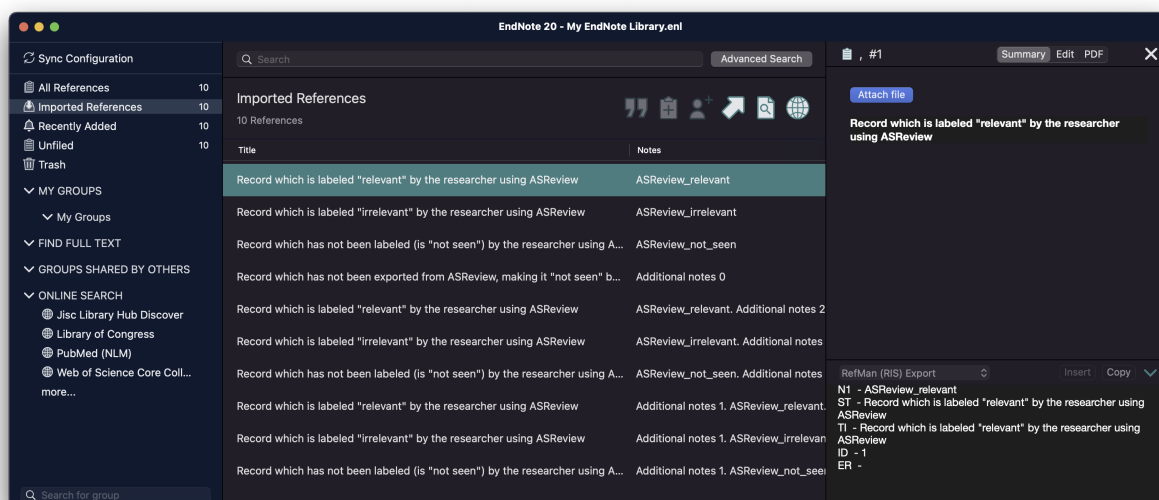


Fig. 2: Example record with a labeling decision imported to Endnote



## FULLY AND PARTIALLY LABELED 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.

---

**Tip:** *Partially labeled data* is useful in the Oracle mode, whereas *Fully labeled data* is useful in the Simulation and Exploration mode.

---

### 9.1 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, or selected by experts to be used for prior knowledge. The value is left empty for the records that you haven’t screened yet, or which are added to the dataset.

For the RIS file format, the labels `ASReview_relevant`, `ASReview_irrelevant`, and `ASReview_not_seen` can be stored with the N1 (Notes) tag. An example of a RIS file with labels in the N1 tag can be found in the [ASReview GitHub repository](#). All labels in this example are valid ways to label the data.

---

**Note:** Exported files containing labeling decisions can be imported into ASReview LAB again, and whereafter all labels are recognized.

---

### 9.2 Partially labeled data

---

**Tip:** Useful for Oracle projects. Read more about *Project modes*.

---

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

A partially labeled dataset can be obtained by exporting results from ASReview LAB or other software. It can also be constructed given the format described above by merging a labeled dataset with new unlabeled records.

Partially labeled datasets are useful as the labels will be recognized by ASReview LAB as *Prior Knowledge*, and labels are used to train the first iteration of the active learning model.

---

**Note:** Merging labeled with unlabeled data should be done outside ASReview LAB, for example, with *Citation Managers*.

---

## 9.3 Fully labeled data

---

**Tip:** Useful for Simulation and Exploration projects. Read more about *Project modes*.

---

Fully labeled datasets are datasets with a labeling decision for each record in the dataset. Fully labeled datasets are useful for exploration or simulation purposes (see also *What is a simulation?* and *Project modes*).

### 9.3.1 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”.

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`.

## 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, Exploration, 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

### 10.1 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*.

#### 10.1.1 Project modes

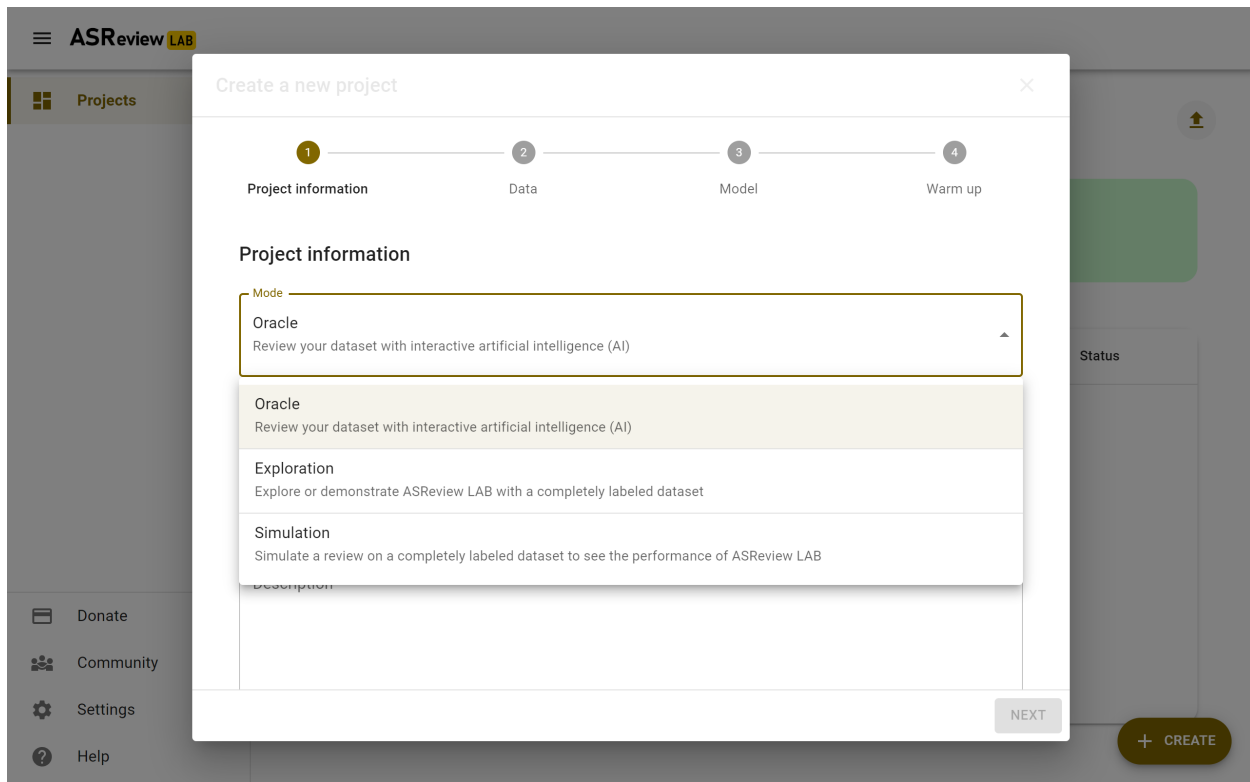
In this step, you have to select a mode. The default is **Oracle**. Oracle mode is used to screen an unlabeled dataset with the help of AI (it's fine if you already have labels, these will be used as prior knowledge). The other two modes, Simulation and Exploration, require fully labeled datasets. They are useful for teaching purposes or studying the performance of active learning in a simulation study.

In short:

- You have an Unlabeled, or *Partially labeled data* -> Oracle
- You have a *Fully labeled data* -> Simulation or Exploration.

#### 10.1.2 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.



## 10.2 Data

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

### 10.2.1 Add Dataset

Click on *Add* to select a dataset. The data needs to adhere to a *specific format*.

Depending on the *Project mode*, you are offered the following options for adding a dataset. Keep in mind that in Oracle mode, your dataset is unlabeled or *Partially labeled data*. For Exploration and Simulation mode, you need *Fully labeled data*.

---

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

---

#### From File

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

### 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

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

### Benchmark Datasets

For Simulation and Exploration only. Select one of the [Benchmark datasets](#). 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.

---

## 10.2.2 Select Prior Knowledge

---

**Note:** If you use [Partially labeled data](#) you can skip this step.

---

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.

To facilitate prior selection, it is possible to search within your dataset. 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 documents. 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.

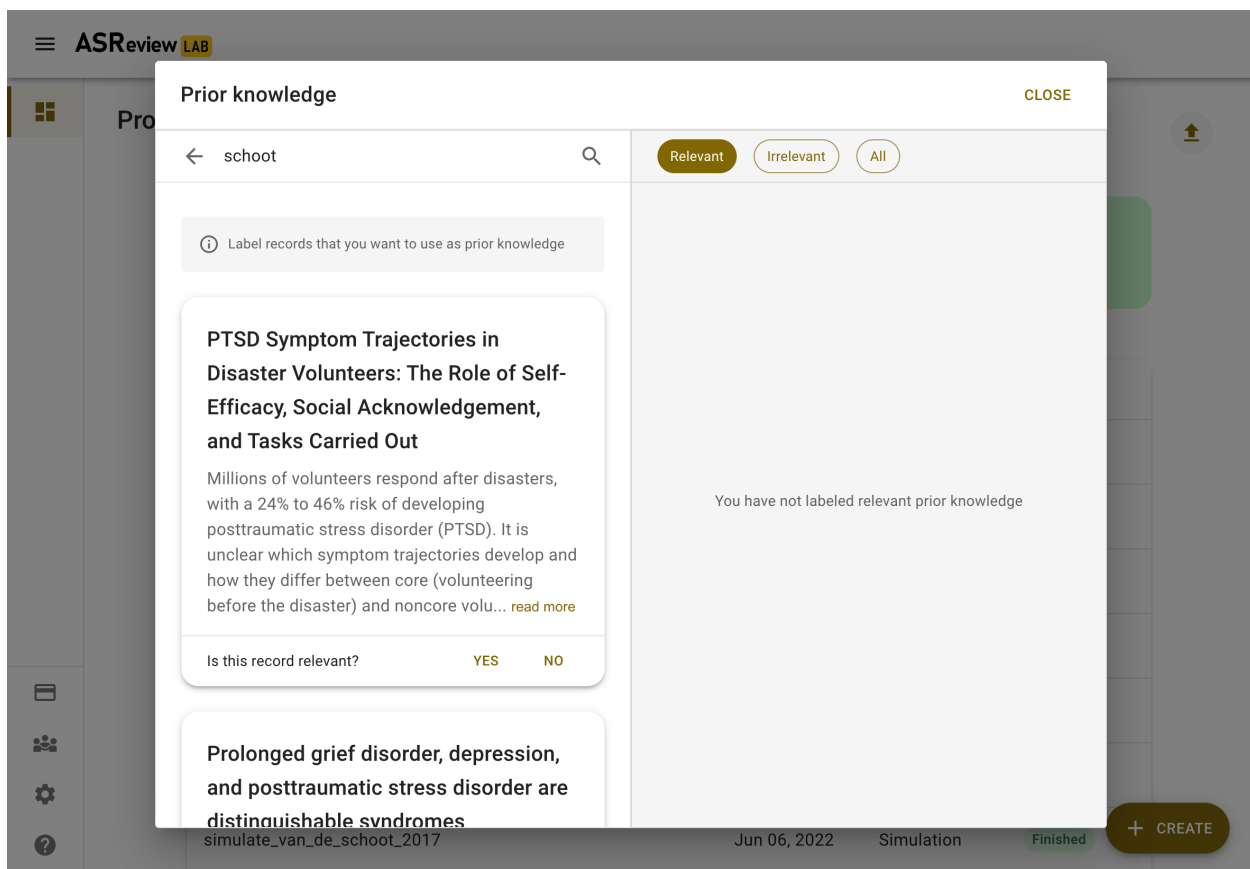
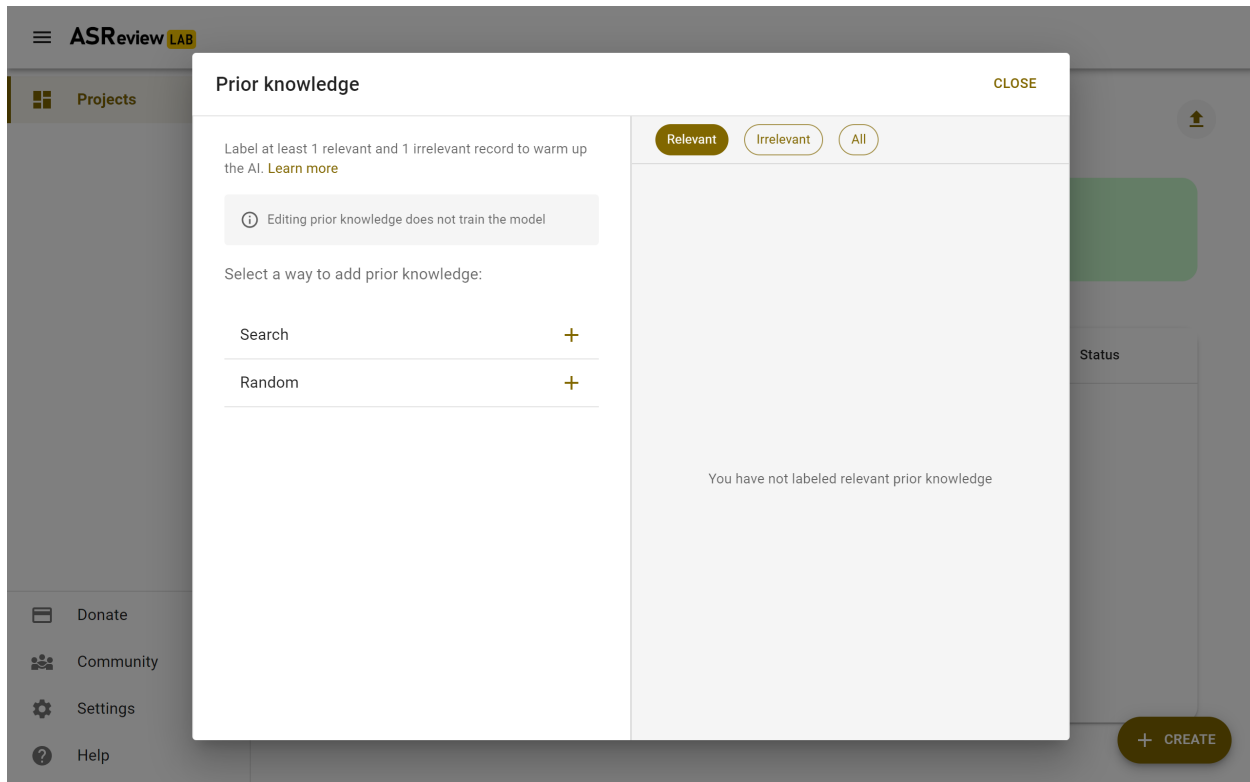
### Search

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).

If you are done searching prior knowledge, click *Close*.





**Prior knowledge** Saved CLOSE

← school 🔍

Relevant (1) Irrelevant All

📘 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](#)

🔖

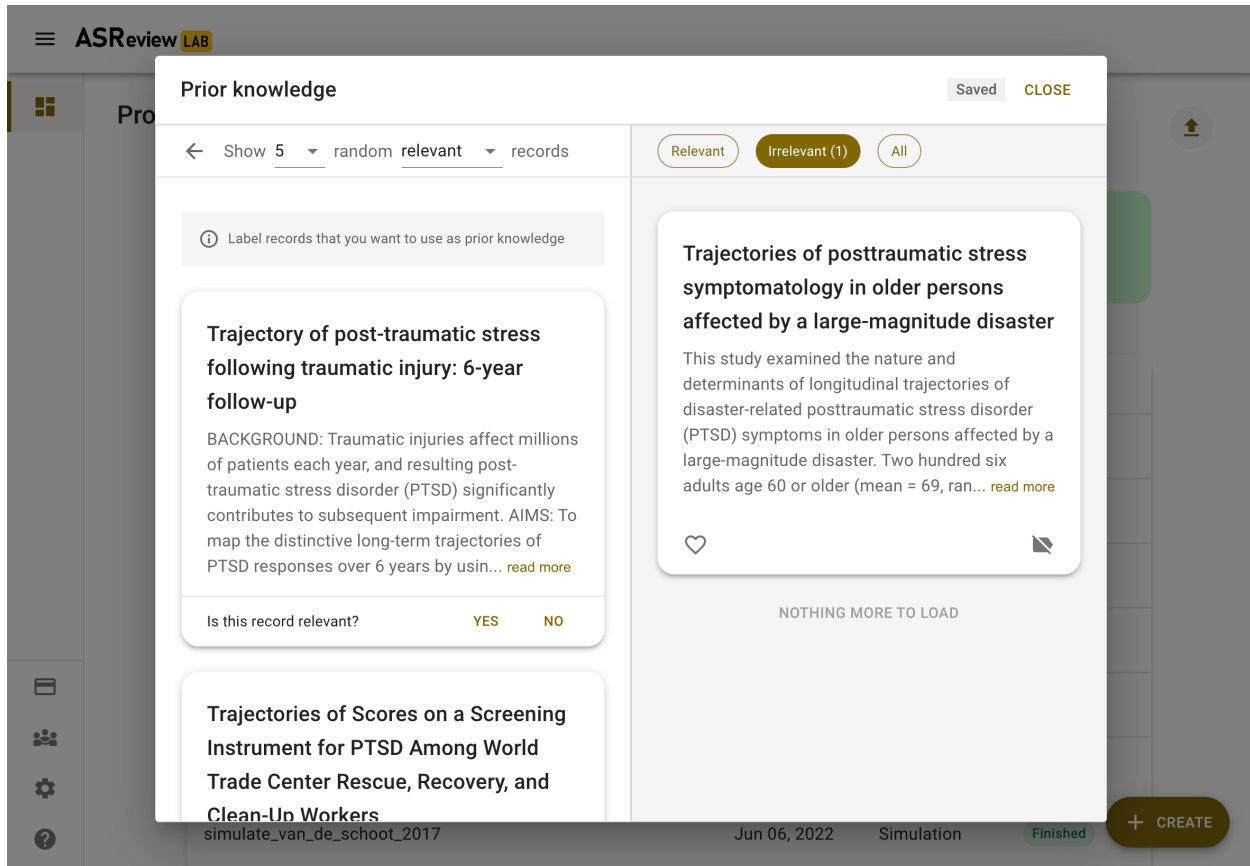
NOTHING MORE TO LOAD

simulate\_van\_de\_schoot\_2017 Jun 06, 2022 Simulation Finished + CREATE

## 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).



The prior knowledge will now show up on the right. Use the buttons to see all prior knowledge or irrelevant items. 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*.

If you are done, click *Close*.

## 10.3 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](#),

### 10.3.1 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 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 an [Model 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 an [Model Extensions](#).

The neural nets require [tensorflow](#), use

```
pip install 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 an [Model 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 an [Model 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.

---

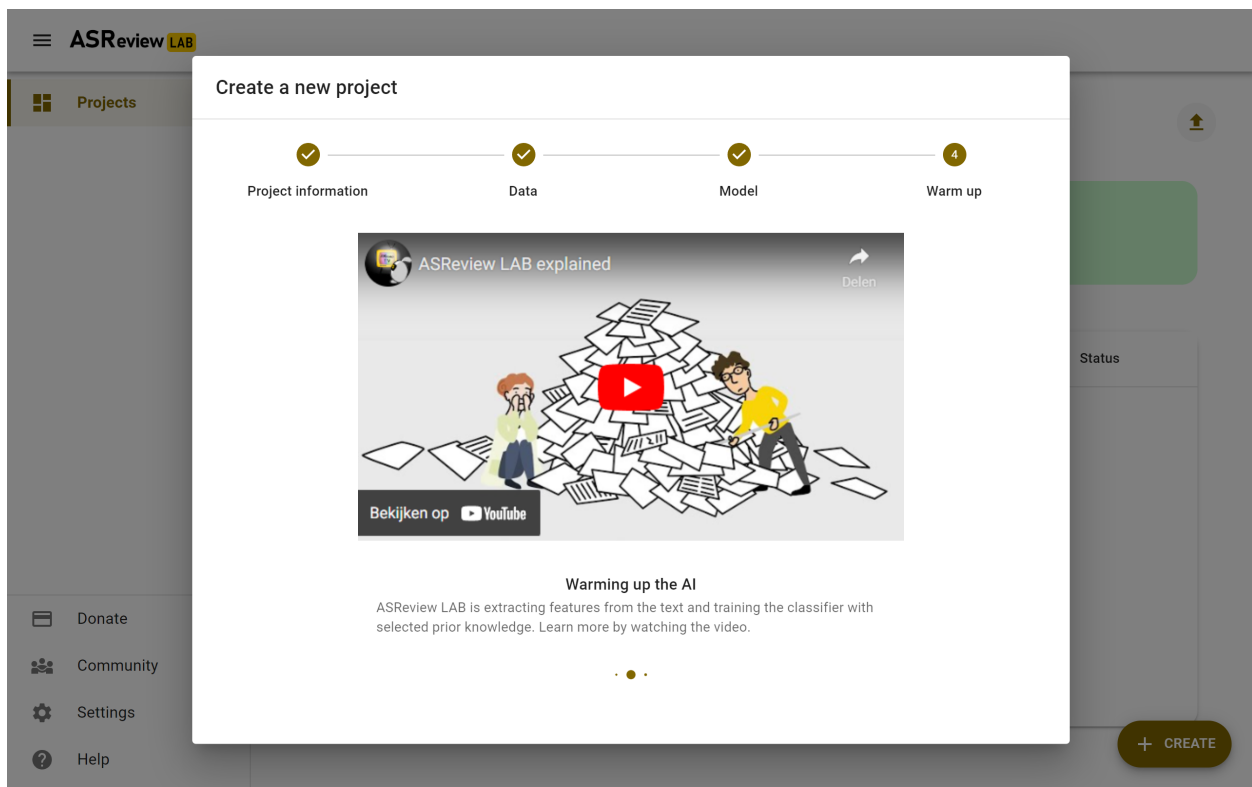
## 10.4 Warm up

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)

---





## SCREENING

---

**Note:** Only for Oracle and Exploration projects. Read more about [Project modes](#).

---


### 11.1 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?

ASReview LAB

← Projects



Your project

Example of Exploration mode

Analytics

**Review**

History

Export

Details



Donate

Community

Settings

Help

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.

ADD NOTE

IRRELEVANT

RELEVANT

---

43

### 11.1.1 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.

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)

---

### 11.1.2 Screening in Exploration mode

The Exploration mode is useful for teaching purposes because a blue bar is displayed on top of the record indicating whether the record has been labeled relevant or irrelevant in the dataset. You can make the same labeling decision without the need to be the oracle.

## 11.2 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).


## 11.3 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.



ASReview LAB

← Projects



Your project

Example of Exploration mode

Analytics

**Review**

History

Export

Details

Donate



Community

Settings

Help

Labeled as relevant in the dataset

## 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.

ADD NOTE

IRRELEVANT

RELEVANT

11.3. Change decisions

45

### 11.3.1 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.

### 11.3.2 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.

ASReview LAB

← Projects

**History**

Relevant Irrelevant All

Filter

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

doi

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 (... [read more](#))

ADD NOTE

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

doi

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.

## 11.4 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.

 Labeled as irrelevant in the dataset

### World Trade Center disaster and A longitudinal study of disaster re



Purpose: The current study examined the role of  
spent working on the site, dust cloud exposure, a

## 11.5 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	<b>r</b> or <b>Shift + r</b>
Label record as irrelevant	<b>i</b> or <b>Shift + i</b>
Return to previous decision	<b>u</b> or <b>Shift + u</b>

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

## 11.6 Display

### 11.6.1 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.

---

### 11.6.2 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.

## 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.

### 12.1 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.

#### 12.1.1 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.

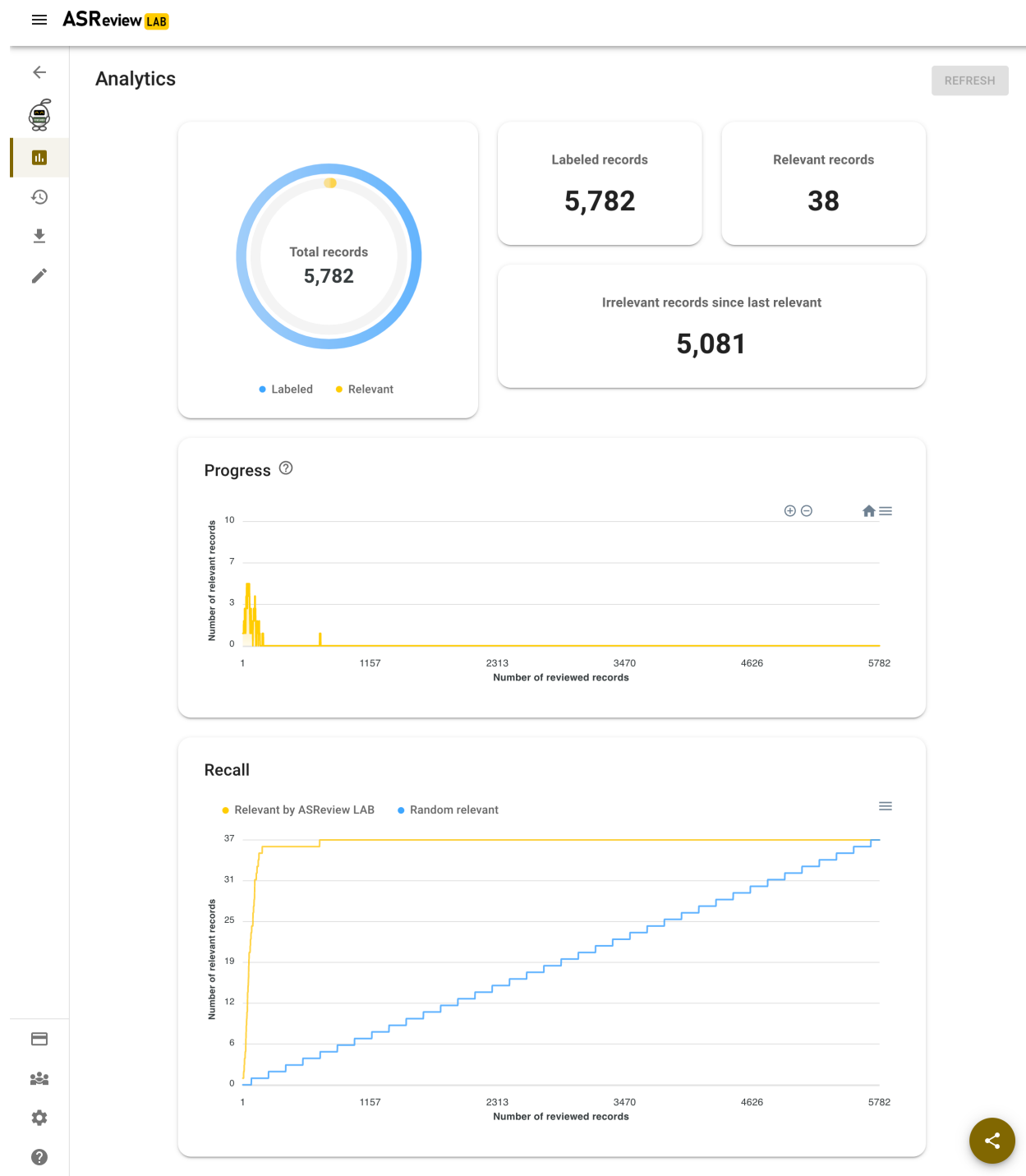


Fig. 1: This figure shows the Analytics page of a fully labeled dataset. All relevant records are found in the first part of the screening.

## 12.1.2 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.

## 12.2 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>`\\_](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.

---

## 12.3 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*.

## 12.4 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.

The following variables will be added to your 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 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.

---




ASReview

LAB

←

Projects



Your project

simulate\_van\_de\_schoot\_2017

Analytics

History

Export

Details

Donate

Community

Settings

Help

Export

Dataset

With relevant/irrelevant labels

File format

CSV (UTF-8)

TSV (UTF-8)

RIS

Available only if you imported a RIS file when creating the project

Excel

12.4. Export results

53



## 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.

≡

ASReview LAB

Projects

Donate

Community




Settings

Help

Projects dashboard

1 Project in Review

1 Project Finished

Project		Date	Mode	Status
simulate_van_de_schoot_2017	  	Jun 06, 2022	Simulation	Finished
Example of Exploration mode		22	Exploration	In Review

Projects per page: 15 1-2 of 2

+ CREATE

## 13.1 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*.

## 13.2 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`).

## 13.3 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.

## EXTENSIONS

ASReview has extensive support for extensions. 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 *Create extensions* for detailed instructions.

### 14.1 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
```

### 14.2 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.

## 14.3 List of extensions for ASReview LAB

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. These extensions can extend the software with new models, subcommands, and datasets.

## 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.

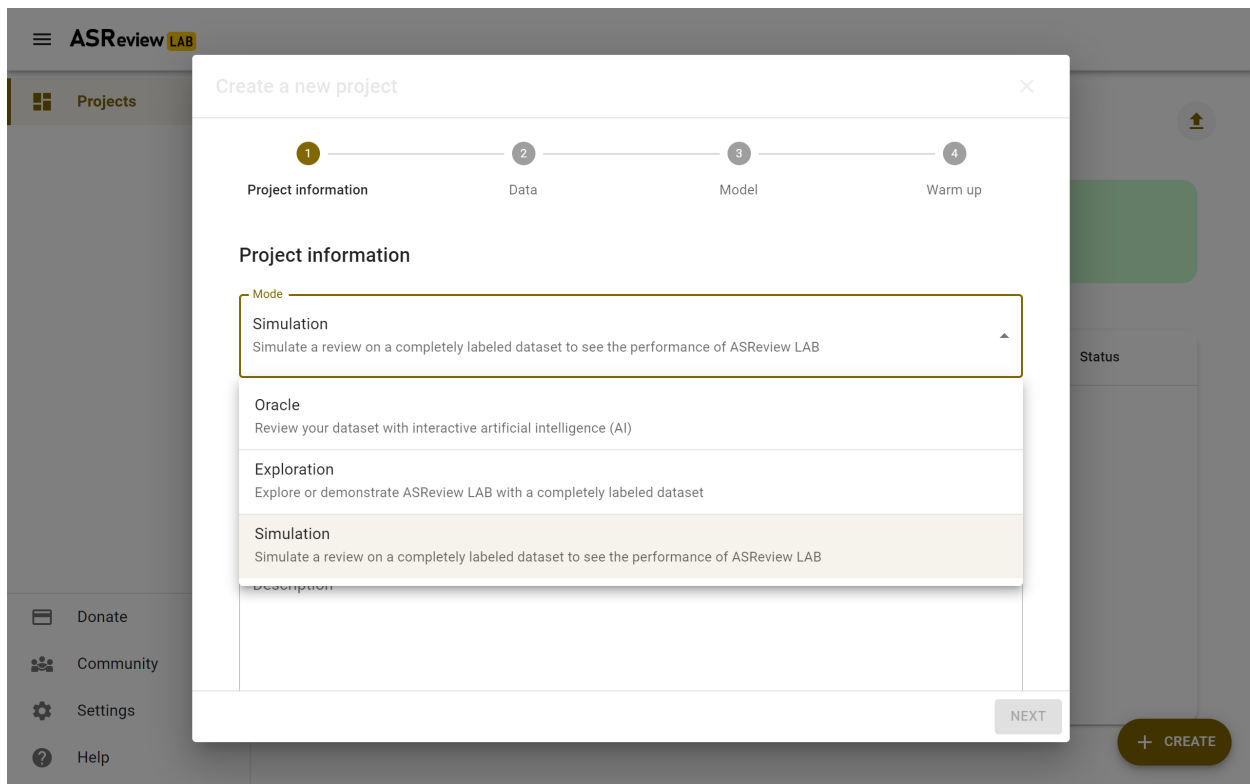
---



## 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).

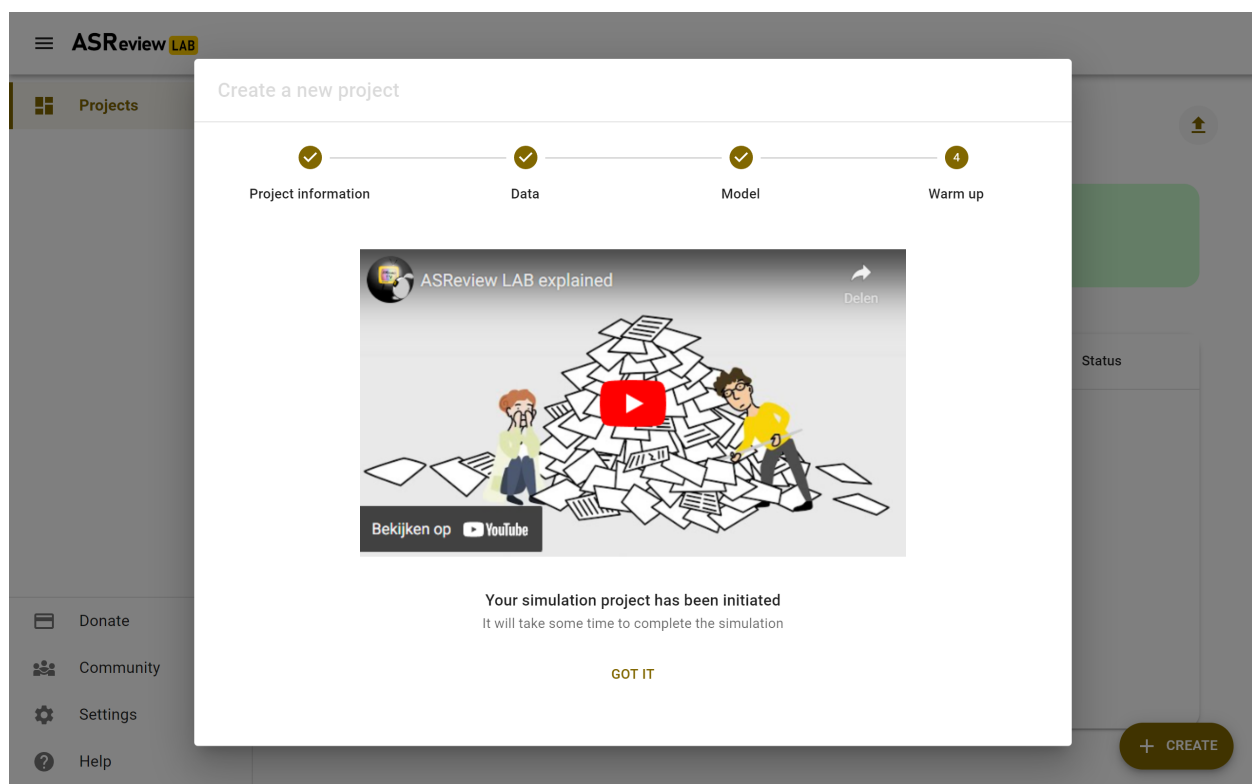
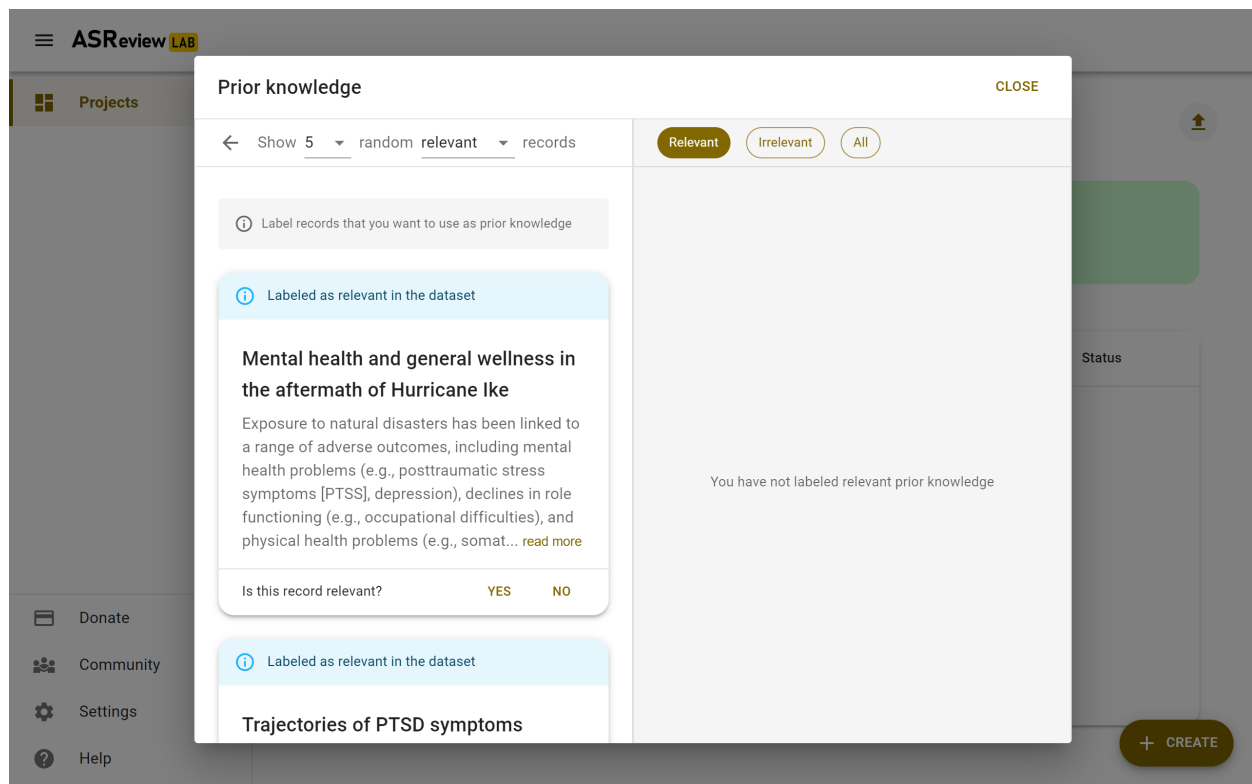


In the step *Data*, import a [fully labeled dataset](#) or use one of the benchmark datasets.

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* differs slightly from the Oracle and Exploration 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 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.



## 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` (AS-Review 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.

#### **--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 *Create 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*.



## 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.





## 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_
↳ 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()
    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
```

(continues on next page)

(continued from previous page)

```
shutil.rmtree(project_path)
```

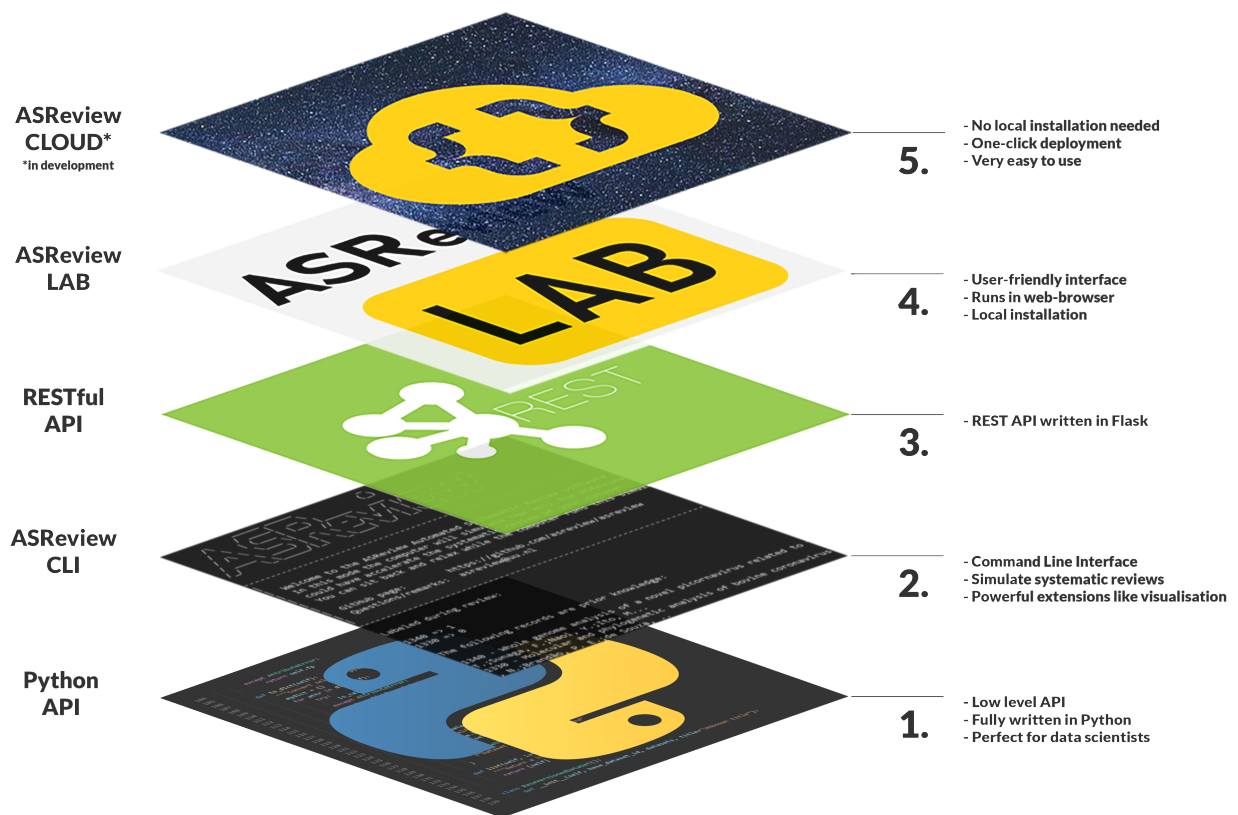


## OVERVIEW

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.

### 20.1 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:



- 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*.

## 20.2 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*.

## 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.





## 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.

### 22.1 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, Exploration, and Simulation mode.

### 22.2 Python Imports

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

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

### 22.3 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': '1.0+8.gf67a5bb',
      'id': 'example',
      'mode': 'simulate',
      'name': 'example',
      'description': 'Simulation created via ASReview via command line interface',
      'authors': None,
      'created_at_unix': 1655364726,
      'datetimeCreated': '2022-06-16 09:32:06.525320',
      'reviews': [{'id': '1654e80a4cbc4bf39e1ea449581fa495',
                    'start_time': '2022-06-16 09:32:06.807452',
                    'status': 'finished',
                    'end_time': '2022-06-16 09:32:14.092992'}],
      '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]:
```

record_id	title \
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...

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

record_id	authors	year	date \
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

(continues on next page)

(continued from previous page)

record_id	doi	label_included	label_abstract_screening	duplicate_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
```

## 22.4 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 566 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         4104        1      None          prior              None
1          457        0      None          prior              None
2         3704        0        nb            max            double
3         2692        0        nb            max            double
4          711        0        nb            max            double

   feature_extraction  training_set  labeling_time  notes
0              None          -1  2022-06-16 09:32:08.461195  None
1              None          -1  2022-06-16 09:32:08.461195  None
2             tfidf           2  2022-06-16 09:32:08.553967  None
3             tfidf           3  2022-06-16 09:32:08.568225  None
4             tfidf           4  2022-06-16 09:32:08.578838  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]:
```

record_id	title \
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 ...

record_id	abstract \
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	

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

record_id	authors	year	date \
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

(continues on next page)

(continued from previous page)

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

record_id	doi	label_included	label_abstract_screening	duplicate_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	

record_id	label	classifier	query_strategy	balance_strategy	\
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	NaN	NaN	NaN	NaN	
6187	NaN	NaN	NaN	NaN	
6188	0.0	nb	max	double	

record_id	feature_extraction	training_set	labeling_time	notes	\
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	NaN	NaN	NaN	NaN	
6187	NaN	NaN	NaN	NaN	
6188	tfidf	324.0	2022-06-16 09:32:11.322652	None	

record_id	labeling_order
-----------	----------------

(continues on next page)

(continued from previous page)

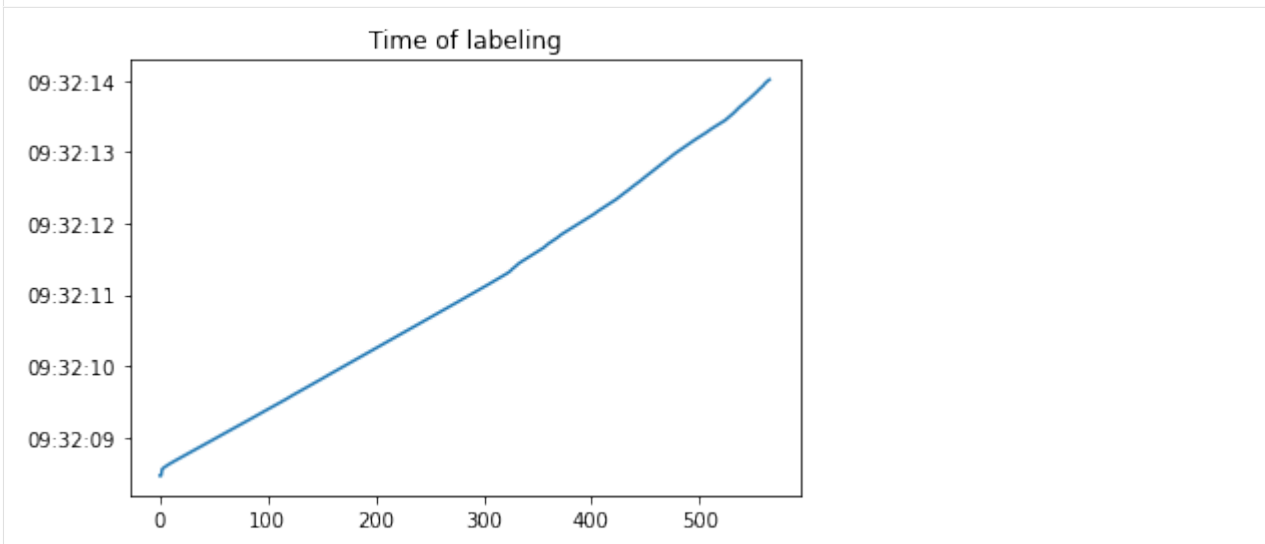
```
record_id
0          NaN
1          NaN
2          NaN
3          NaN
4          NaN
...        ...
6184       NaN
6185       NaN
6186       NaN
6187       NaN
6188       324.0
```

```
[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]: <AxesSubplot: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      0
      1      0
      2      0
      3      0
      4      0
```

(continues on next page)

(continued from previous page)

```

..
559    0
560    0
561    0
562    0
563    1
Name: label, Length: 564, 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)
      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              63  2022-06-16 09:32:09.083643  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': '1.0+8.gf67a5bb',
      'model_has_trained': True}

```

## 22.5 Cleanup

The following code removes the temporary folder that was created:

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



## CREATE 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 [List of extensions for ASReview LAB](#)? Create a Pull Request or open an issue on [GitHub](#).

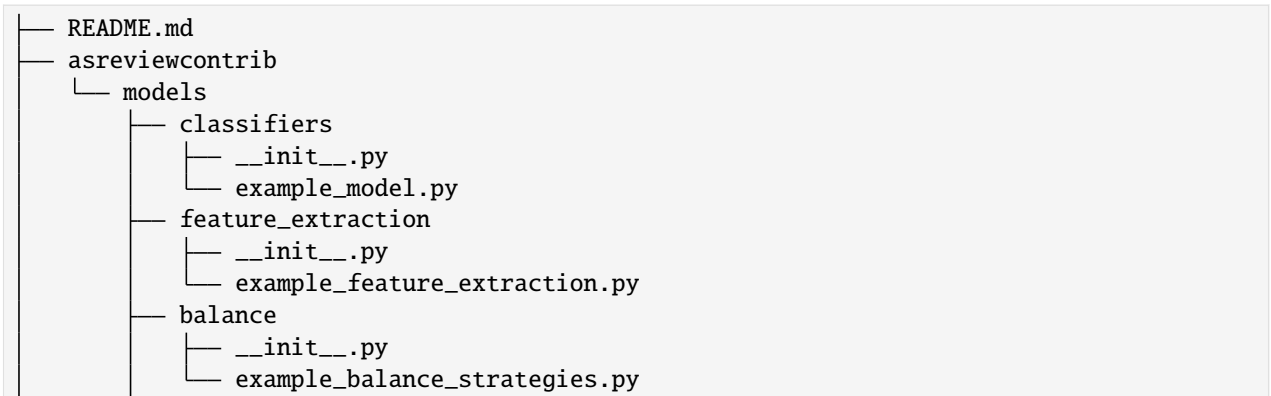
For more information on the ASReview API for creating an extension, a technical reference for development is found under the [API reference](#). This technical reference contains functions for use in your extension, and an overview of all classes to extend on.

### 23.1 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 `asreviewcontrib` package. 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:



(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`.

## 23.2 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.

## 23.3 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`.



## API REFERENCE

### 24.1 Data and datasets

#### 24.1.1 Read data

<code>load_data(name, *args, **kwargs)</code>	Load data from file, URL, or plugin.
<code>ASReviewData([df, column_spec])</code>	Data object to the dataset with texts, labels, DOIs etc.

##### **asreview.load\_data**

`asreview.load_data(name, *args, **kwargs)`

Load data from file, URL, or plugin.

##### **Parameters**

**name** (*str*, *pathlib.Path*) – File path, URL, or alias of extension dataset.

##### **Returns**

*asreview.ASReviewData* – Inititalized ASReview data object.

##### **asreview.ASReviewData**

`class asreview.ASReviewData(df=None, column_spec=None)`

Data 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

<i>abstract</i>	
<i>authors</i>	
<i>bodies</i>	
<i>doi</i>	
<i>final_included</i>	
<i>headings</i>	
<i>included</i>	
<i>keywords</i>	
<i>labels</i>	
<i>notes</i>	
<i>prior_data_idx</i>	Get prior_included, prior_excluded from dataset.
<i>record_ids</i>	
<i>texts</i>	
<i>title</i>	
<i>url</i>	

**asreview.ASReviewData.abstract**

**property** ASReviewData.**abstract**

**asreview.ASReviewData.authors**

**property** ASReviewData.**authors**

**asreview.ASReviewData.bodies**

**property** ASReviewData.**bodies**

**asreview.ASReviewData.doi**

**property** ASReviewData.**doi**

**asreview.ASReviewData.final\_included**

**property** ASReviewData.**final\_included**

**asreview.ASReviewData.headings**

**property** ASReviewData.**headings**

**asreview.ASReviewData.included**

**property** ASReviewData.**included**

**asreview.ASReviewData.keywords**

**property** ASReviewData.**keywords**

**asreview.ASReviewData.labels**

**property** ASReviewData.**labels**

**asreview.ASReviewData.notes**

**property** ASReviewData.notes

**asreview.ASReviewData.prior\_data\_idx**

**property** ASReviewData.prior\_data\_idx

Get prior\_included, prior\_excluded from dataset.

**asreview.ASReviewData.record\_ids**

**property** ASReviewData.record\_ids

**asreview.ASReviewData.texts**

**property** ASReviewData.texts

**asreview.ASReviewData.title**

**property** ASReviewData.title

**asreview.ASReviewData.url**

**property** ASReviewData.url

**Methods**

<i>drop_duplicates</i> ([pid, inplace, reset_index])	Drop duplicate records.
<i>duplicated</i> ([pid])	Return boolean Series denoting duplicate rows.
<i>from_extension</i> (name[, reader])	Load a dataset from extension.
<i>from_file</i> (fp[, reader])	Create instance from supported file format.
<i>get</i> (name)	Get column with name.
<i>hash</i> ()	Compute a hash from the dataset.
<i>prior_labels</i> (state[, by_index])	Get the labels that are marked as 'prior'.
<i>record</i> (i[, by_index])	Create a record from an index.
<i>to_dataframe</i> ([labels, ranking])	Create new dataframe with updated label (order).
<i>to_file</i> (fp[, labels, ranking, writer])	Export data object to file.



## asreview.ASReviewData.drop\_duplicates

ASReviewData.**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.ASReviewData.duplicated

ASReviewData.**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.ASReviewData.from\_extension

**classmethod** ASReviewData.**from\_extension**(name, reader=None)

Load a dataset from extension.

### Parameters

- **fp** (*str*, *pathlib.Path*) – Read the data from this file or url.
- **reader** (*class*) – Reader to import the file.

## asreview.ASReviewData.from\_file

**classmethod** ASReviewData.**from\_file**(fp, reader=None)

Create instance from supported file format.

It works in two ways; either manual control where the conversion functions are supplied or automatic, where it searches in the entry points for the right conversion functions.

### Parameters

- **fp** (*str*, *pathlib.Path*) – Read the data from this file or url.

- **reader** (*class*) – Reader to import the file.

### **asreview.ASReviewData.get**

ASReviewData.**get**(*name*)

Get column with name.

### **asreview.ASReviewData.hash**

ASReviewData.**hash**()

Compute a hash from the dataset.

#### **Returns**

*str* – SHA1 hash, computed from the titles/abstracts of the dataframe.

### **asreview.ASReviewData.prior\_labels**

ASReviewData.**prior\_labels**(*state*, *by\_index=True*)

Get the labels that are marked as ‘prior’.

#### **state: BaseState**

Open state that contains the label information.

#### **by\_index: bool**

If True, return internal indexing. If False, return `record_ids` for indexing.

#### **Returns**

*numpy.ndarray* – Array of indices that have the ‘prior’ property.

### **asreview.ASReviewData.record**

ASReviewData.**record**(*i*, *by\_index=True*)

Create a record from an index.

#### **Parameters**

- **i** (*int*, *iterable*) – Index of the record, or list of indices.
- **by\_index** (*bool*) – If True, take the *i*-th value as used internally by the review. If False, take the record with `record_id==i`.

#### **Returns**

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

**asreview.ASReviewData.to\_dataframe**

`ASReviewData.to_dataframe(labels=None, ranking=None)`

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.

**Returns**

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

**asreview.ASReviewData.to\_file**

`ASReviewData.to_file(fp, labels=None, ranking=None, writer=None)`

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.

**24.1.2 Statistics**

<code>data.statistics.abstract_length(data)</code>	Return the average length of the abstracts.
<code>data.statistics.n_duplicates(data[, pid])</code>	Number of duplicates.
<code>data.statistics.n_irrelevant(data)</code>	Return the number of irrelevant records.
<code>data.statistics.n_keywords(data)</code>	Return the number of keywords.
<code>data.statistics.n_missing_abstract(data)</code>	Return the number of records with missing abstracts.
<code>data.statistics.n_missing_title(data)</code>	Return the number of records with missing titles.
<code>data.statistics.n_records(data)</code>	Return the number of records.
<code>data.statistics.n_relevant(data)</code>	Return the number of relevant records.
<code>data.statistics.n_unlabeled(data)</code>	Return the number of unlabeled records.
<code>data.statistics.title_length(data)</code>	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.data.ASReviewData`) – An `ASReviewData` 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.data.ASReviewData`) – An `ASReviewData` 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.data.ASReviewData`) – An `ASReviewData` 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.data.ASReviewData`) – An `ASReviewData` 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.data.ASReviewData*) – An ASReviewData 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.data.ASReviewData*) – An ASReviewData 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.data.ASReviewData*) – An ASReviewData 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.data.ASReviewData*) – An ASReviewData 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.data.ASReviewData*) – An ASReviewData 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.data.ASReviewData*) – An ASReviewData object with the records.**Returns***int* – The statistic

## 24.1.3 Datasets

**Available datasets**

<code>asreview.datasets.SynergyDataGroup()</code>	Datasets available in the SYNERGY dataset.
<code>asreview.datasets. NaturePublicationDataGroup()</code>	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**

<code>description</code>
<code>group_id</code>

**asreview.datasets.SynergyDataGroup.description**`SynergyDataGroup.description = 'SYNERGY datasets (asreview.ai/synergy)'`**asreview.datasets.SynergyDataGroup.group\_id**`SynergyDataGroup.group_id = 'synergy'`

## Methods

<code>append(dataset)</code>	Append dataset to group.
<code>find(dataset_id)</code>	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

<code>description</code>
<code>group_id</code>

### `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**

<code>append(dataset)</code>	Append dataset to group.
<code>find(dataset_id)</code>	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=[], **kwargs)
```

**Attributes**

<i>filename</i>
<i>reader</i>

**asreview.datasets.BaseDataSet.filename**

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

**asreview.datasets.BaseDataSet.reader**

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

**Methods**

<i>to_file</i> (path)
-----------------------

**asreview.datasets.BaseDataSet.to\_file**

BaseDataSet.**to\_file**(path)

**asreview.datasets.BaseDataGroup**

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

## Attributes

<i>description</i>
<i>group_id</i>

**asreview.datasets.BaseDataGroup.description**

**abstract property** BaseDataGroup.description

**asreview.datasets.BaseDataGroup.group\_id**

**abstract property** BaseDataGroup.group\_id

## Methods

<i>append</i> (dataset)	Append dataset to group.
<i>find</i> (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**

<i>groups</i>
---------------

**asreview.datasets.DatasetManager.groups**

```
property DatasetManager.groups
```

**Methods**

<i>find</i> (dataset_id)	Find a dataset.
<i>list</i> ([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.

## 24.2 Reviewer

<code>review.BaseReview(as_data, project[, model, ...])</code>	Base class for Systematic Review.
<code>review.ReviewSimulate(as_data, *args[, ...])</code>	ASReview Simulation mode class.

### 24.2.1 asreview.review.BaseReview

```
class asreview.review.BaseReview(as_data, project,
                                model=<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_papers=None, n_instances=1, stop_if=None, start_idx=[])
```

Base class for Systematic Review.

#### Parameters

- **as\_data** (`asreview.ASReviewData`) – The data object which contains the text, labels, etc.
- **project** (*path-like*) – Path to the project file.
- **model** (`BaseTrainClassifier`) – Initialized model to fit the data during active learning. See `asreview.models.utils.py` for possible models.
- **query\_model** (`BaseQueryStrategy`) – 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** (`BaseBalance`) – Initialized model to redistribute the training data during the active learning process. They might either resample or undersample specific papers.
- **feature\_model** (`BaseFeatureExtraction`) – Feature extraction model that converts texts and keywords to feature matrices.
- **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.

#### Attributes

<code>settings</code>	Get an ASReview settings object
-----------------------	---------------------------------

**asreview.review.BaseReview.settings****property** BaseReview.settings

Get an ASReview settings object

**Methods**

<code>review()</code>	Do a full review.
<code>train()</code>	Train a new model on the labeled data.

**asreview.review.BaseReview.review****BaseReview.review()**

Do a full review.

**asreview.review.BaseReview.train****BaseReview.train()**

Train a new model on the labeled data.

**24.2.2 asreview.review.ReviewSimulate**

```
class asreview.review.ReviewSimulate(as_data, *args, n_prior_included=0, n_prior_excluded=0,
                                     prior_indices=None, init_seed=None, write_interval=None,
                                     **kwargs)
```

ASReview Simulation mode class.

**Parameters**

- **as\_data** (`asreview.ASReviewData`) – 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

<i>name</i>	
<i>settings</i>	Get an ASReview settings object

### `asreview.review.ReviewSimulate.name`

`ReviewSimulate.name = 'simulate'`

### `asreview.review.ReviewSimulate.settings`

**property** `ReviewSimulate.settings`

Get an ASReview settings object

## Methods

<i>review()</i>	Do a full review.
<i>train()</i>	Train a new model on the labeled data.

### `asreview.review.ReviewSimulate.review`

`ReviewSimulate.review()`

Do a full review.

### `asreview.review.ReviewSimulate.train`

`ReviewSimulate.train()`

Train a new model on the labeled data.

## 24.3 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

<code>models.base.BaseModel()</code>	Abstract class for any kind of model.
--------------------------------------	---------------------------------------

### 24.3.1 asreview.models.base.BaseModel

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

Abstract class for any kind of model.

#### Attributes

<code>default_param</code>	Get the default parameters of the model.
<code>name</code>	
<code>param</code>	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

<code>full_hyper_space()</code>
<code>hyper_space()</code>

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

`BaseModel.full_hyper_space()`

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

`BaseModel.hyper_space()`

## 24.3.2 `asreview.models.feature_extraction`

### Classes

<code>feature_extraction.base.BaseFeatureExtraction([...])</code>	Base class for feature extraction methods.
<code>feature_extraction.Tfidf(*args[, ngram_max, ...])</code>	TF-IDF feature extraction technique (tfidf).
<code>feature_extraction.Doc2Vec(*args[, ...])</code>	Doc2Vec feature extraction technique (doc2vec).
<code>feature_extraction.EmbeddingIdf(*args[, ...])</code>	Embedding IDF feature extraction technique (embedding-idf).
<code>feature_extraction.EmbeddingLSTM(*args[, ...])</code>	Embedding LSTM feature extraction technique (embedding-lstm).
<code>feature_extraction.SBERT(*args[, ...])</code>	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

<code>default_param</code>	Get the default parameters of the model.
<code>name</code>	
<code>param</code>	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**

<i>fit</i> (texts)	Fit the model to the texts.
<i>fit_transform</i> (texts[, titles, abstracts, ...])	Fit and transform a list of texts.
<i>full_hyper_space</i> ()	
<i>hyper_space</i> ()	
<i>transform</i> (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

<code>default_param</code>	Get the default parameters of the model.
<code>label</code>	
<code>name</code>	
<code>param</code>	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

<code>fit(texts)</code>	Fit the model to the texts.
<code>fit_transform(texts[, titles, abstracts, ...])</code>	Fit and transform a list of texts.
<code>full_hyper_space()</code>	
<code>hyper_space()</code>	
<code>transform(texts)</code>	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 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 occurrences 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

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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**

<i>fit</i> (texts)	Fit the model to the texts.
<i>fit_transform</i> (texts[, titles, abstracts, ...])	Fit and transform a list of texts.
<i>full_hyper_space</i> ()	
<i>hyper_space</i> ()	
<i>transform</i> (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 tensorflow` or install all optional ASReview dependencies with `pip install asreview[all]`

---

**Parameters**

**embedding\_fp** (*str*) – Path to embedding.

**Attributes**

<i>default_param_label</i>	Get the default parameters of the model.
<i>name</i>	
<i>param</i>	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**

<i>fit</i> (texts)	Fit the model to the texts.
<i>fit_transform</i> (texts[, titles, abstracts, ...])	Fit and transform a list of texts.
<i>full_hyper_space</i> ()	
<i>hyper_space</i> ()	
<i>transform</i> (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 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 truncated. 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

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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

<code>fit(texts)</code>	Fit the model to the texts.
<code>fit_transform(texts[, titles, abstracts, ...])</code>	Fit and transform a list of texts.
<code>full_hyper_space()</code>	
<code>get_embedding_matrix(texts, embedding_fp)</code>	
<code>hyper_space()</code>	
<code>transform(texts)</code>	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 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

<code>default_param</code>	Get the default parameters of the model.
<code>label</code>	
<code>name</code>	
<code>param</code>	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

<code>fit(texts)</code>	Fit the model to the texts.
<code>fit_transform(texts[, titles, abstracts, ...])</code>	Fit and transform a list of texts.
<code>full_hyper_space()</code>	
<code>hyper_space()</code>	
<code>transform(texts)</code>	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

<i>feature_extraction.get_feature_model</i> (name, *args)	Get an instance of a feature extraction model from a string.
<i>feature_extraction.get_feature_class</i> (name)	Get class of feature extraction from string.
<i>feature_extraction.list_feature_extraction</i> ()	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 the feature 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.

**24.3.3 asreview.models.classifiers**

Classes

<code>classifiers.base.BaseTrainClassifier()</code>	Base model, abstract class to be implemented by derived ones.
<code>classifiers.NaiveBayesClassifier([alpha])</code>	Naive Bayes classifier (nb).
<code>classifiers.RandomForestClassifier([...])</code>	Random forest classifier (rf).
<code>classifiers.SVMClassifier([gamma, ...])</code>	Support vector machine classifier (svm).
<code>classifiers.LogisticClassifier([C, ...])</code>	Logistic regression classifier (logistic).
<code>classifiers.LSTMBaseClassifier([...])</code>	LSTM-base classifier (lstm-base).
<code>classifiers.LSTMPoolClassifier([...])</code>	LSTM-pool classifier (lstm-pool).
<code>classifiers.NN2LayerClassifier([...])</code>	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. All functions dealing with hyperparameters can be ignore if you don't use hyperopt for hyperparameter tuning. 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**

<i>default_param</i>	Get the default parameters of the model.
<i>name</i>	
<i>param</i>	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**

<i>fit</i> (X, y)	Fit the model to the data.
<i>full_hyper_space</i> ()	Get a hyperparameter space to use with hyperopt.
<i>hyper_space</i> ()	
<i>predict_proba</i> (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**()

Get a hyperparameter space to use with hyperopt.

**Returns**

*dict, dict* – Parameter space. Parameter choices; in case of hyperparameters with a list of choices, store the choices there.

**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

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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

<i>fit(X, y)</i>	Fit the model to the data.
<i>full_hyper_space()</i>	Get a hyperparameter space to use with hyperopt.
<i>hyper_space()</i>	
<i>predict_proba(X)</i>	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()`

Get a hyperparameter space to use with hyperopt.

**Returns**

*dict, dict* – Parameter space. Parameter choices; in case of hyperparameters with a list of choices, store the choices there.

**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 RandomState, 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

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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

<i>fit(X, y)</i>	Fit the model to the data.
<i>full_hyper_space()</i>	Get a hyperparameter space to use with hyperopt.
<i>hyper_space()</i>	
<i>predict_proba(X)</i>	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()`

Get a hyperparameter space to use with hyperopt.

**Returns**

*dict, dict* – Parameter space. Parameter choices; in case of hyperparameters with a list of choices, store the choices there.

**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, RandomState*) – State of the RNG.

## Attributes

<code>default_param</code>	Get the default parameters of the model.
<code>label</code>	
<code>name</code>	
<code>param</code>	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

<code>fit(X, y)</code>	Fit the model to the data.
<code>full_hyper_space()</code>	Get a hyperparameter space to use with hyperopt.
<code>hyper_space()</code>	
<code>predict_proba(X)</code>	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**()

Get a hyperparameter space to use with hyperopt.

**Returns**

*dict, dict* – Parameter space. Parameter choices; in case of hyperparameters with a list of choices, store the choices there.

**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*, *RandomState*) – Random state for the model.
- **n\_jobs** (*int*) – Number of CPU cores used.

## Attributes

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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

<i>fit(X, y)</i>	Fit the model to the data.
<i>full_hyper_space()</i>	Get a hyperparameter space to use with hyperopt.
<i>hyper_space()</i>	
<i>predict_proba(X)</i>	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()`

Get a hyperparameter space to use with hyperopt.

**Returns**

*dict, dict* – Parameter space. Parameter choices; in case of hyperparameters with a list of choices, store the choices there.

**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.LSTMBaseClassifier**

```
class asreview.models.classifiers.LSTMBaseClassifier(embedding_matrix=None, backwards=True,
                                                    dropout=0.4, optimizer='rmsprop',
                                                    lstm_out_width=20, learn_rate=1.0,
                                                    dense_width=128, verbose=0, batch_size=32,
                                                    epochs=35, shuffle=False, class_weight=30.0)
```

LSTM-base classifier (lstm-base).

LSTM model that consists of an embedding layer, LSTM layer with one output, dense layer, and a single sigmoid output node. Use the `asreview.models.feature_extraction.EmbeddingLSTM` feature extraction method. Currently not so well optimized and slow.

---

**Note:** This model requires `tensorflow` to be installed. Use `pip install tensorflow` or install all optional ASReview dependencies with `pip install asreview[all]`

---

### Parameters

- **embedding\_matrix** (*numpy.ndarray*) – Embedding matrix to use with LSTM model.
- **backwards** (*bool*) – Whether to have a forward or backward LSTM.
- **dropout** (*float*) – Value in [0, 1.0) that gives the dropout and recurrent dropout rate for the LSTM model.
- **optimizer** (*str*) – Optimizer to use.
- **lstm\_out\_width** (*int*) – Output width of the LSTM.
- **learn\_rate** (*float*) – Learn rate multiplier of default learning rate.
- **dense\_width** (*int*) – Size of the dense layer of the model.
- **verbose** (*int*) – Verbosity.
- **batch\_size** (*int*) – Size of the batch size for the LSTM model.
- **epochs** (*int*) – Number of epochs to train the LSTM model.
- **shuffle** (*bool*) – Whether to shuffle the data before starting to train.
- **class\_weight** (*float*) – Class weight for the included papers.

### Attributes

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	Get the (assigned) parameters of the model.

### **asreview.models.classifiers.LSTMBaseClassifier.default\_param**

#### **property** LSTMBaseClassifier.default\_param

Get the default parameters of the model.

#### **Returns**

*dict* – Dictionary with parameter: default value

### **asreview.models.classifiers.LSTMBaseClassifier.label**

LSTMBaseClassifier.label = 'LSTM classic'

**asreview.models.classifiers.LSTMBaseClassifier.name**

LSTMBaseClassifier.name = 'lstm-base'

**asreview.models.classifiers.LSTMBaseClassifier.param**

**property** LSTMBaseClassifier.param

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

**Methods**

<i>fit</i> (X, y)	Fit the model to the data.
<i>full_hyper_space</i> ()	Get a hyperparameter space to use with hyperopt.
<i>hyper_space</i> ()	
<i>predict_proba</i> (X)	Get the inclusion probability for each sample.

**asreview.models.classifiers.LSTMBaseClassifier.fit**

LSTMBaseClassifier.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.LSTMBaseClassifier.full\_hyper\_space**

LSTMBaseClassifier.full\_hyper\_space()

Get a hyperparameter space to use with hyperopt.

**Returns**

*dict, dict* – Parameter space. Parameter choices; in case of hyperparameters with a list of choices, store the choices there.

**asreview.models.classifiers.LSTMBaseClassifier.hyper\_space**`LSTMBaseClassifier.hyper_space()`**asreview.models.classifiers.LSTMBaseClassifier.predict\_proba**`LSTMBaseClassifier.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.LSTMPoolClassifier**

```
class asreview.models.classifiers.LSTMPoolClassifier(embedding_matrix=None, backwards=True,
                                                    dropout=0.4, optimizer='rmsprop',
                                                    lstm_out_width=20, lstm_pool_size=128,
                                                    learn_rate=1.0, verbose=0, batch_size=32,
                                                    epochs=35, shuffle=False, class_weight=30.0)
```

LSTM-pool classifier (lstm-pool).

LSTM model that consists of an embedding layer, LSTM layer with many outputs, max pooling layer, and a single sigmoid output node. Use the `asreview.models.feature_extraction.EmbeddingLSTM` feature extraction method. Currently not so well optimized and slow.

---

**Note:** This model requires `tensorflow` to be installed. Use `pip install tensorflow` or install all optional ASReview dependencies with `pip install asreview[all]`

---

**Parameters**

- **embedding\_matrix** (*numpy.ndarray*) – Embedding matrix to use with LSTM model.
- **backwards** (*bool*) – Whether to have a forward or backward LSTM.
- **dropout** (*float*) – Value in [0, 1.0) that gives the dropout and recurrent dropout rate for the LSTM model.
- **optimizer** (*str*) – Optimizer to use.
- **lstm\_out\_width** (*int*) – Output width of the LSTM.
- **lstm\_pool\_size** (*int*) – Size of the pool, must be a divisor of `max_sequence_length`.
- **learn\_rate** (*float*) – Learn rate multiplier of default learning rate.
- **verbose** (*int*) – Verbosity.
- **batch\_size** (*int*) – Size of the batch size for the LSTM model.
- **epochs** (*int*) – Number of epochs to train the LSTM model.
- **shuffle** (*bool*) – Whether to shuffle the data before starting to train.

- **class\_weight** (*float*) – Class weight for the included papers.

## Attributes

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	Get the (assigned) parameters of the model.

### asreview.models.classifiers.LSTMPoolClassifier.default\_param

**property** LSTMPoolClassifier.**default\_param**

Get the default parameters of the model.

**Returns**

*dict* – Dictionary with parameter: default value

### asreview.models.classifiers.LSTMPoolClassifier.label

LSTMPoolClassifier.**label** = 'LSTM with a max pooling layer'

### asreview.models.classifiers.LSTMPoolClassifier.name

LSTMPoolClassifier.**name** = 'lstm-pool'

### asreview.models.classifiers.LSTMPoolClassifier.param

**property** LSTMPoolClassifier.**param**

Get the (assigned) parameters of the model.

**Returns**

*dict* – Dictionary with parameter: current value.

## Methods

<i>fit</i> (X, y)	Fit the model to the data.
<i>full_hyper_space</i> ()	Get a hyperparameter space to use with hyperopt.
<i>hyper_space</i> ()	
<i>predict_proba</i> (X)	Get the inclusion probability for each sample.

**asreview.models.classifiers.LSTMPoolClassifier.fit**

LSTMPoolClassifier.**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.LSTMPoolClassifier.full\_hyper\_space**

LSTMPoolClassifier.**full\_hyper\_space**()

Get a hyperparameter space to use with hyperopt.

**Returns**

*dict, dict* – Parameter space. Parameter choices; in case of hyperparameters with a list of choices, store the choices there.

**asreview.models.classifiers.LSTMPoolClassifier.hyper\_space**

LSTMPoolClassifier.**hyper\_space**()

**asreview.models.classifiers.LSTMPoolClassifier.predict\_proba**

LSTMPoolClassifier.**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 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

<code>default_param</code>	Get the default parameters of the model.
<code>label</code>	
<code>name</code>	
<code>param</code>	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**

<i>fit</i> (X, y)	Fit the model to the data.
<i>full_hyper_space</i> ()	Get a hyperparameter space to use with hyperopt.
<i>hyper_space</i> ()	
<i>predict_proba</i> (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()

Get a hyperparameter space to use with hyperopt.

**Returns**

*dict, dict* – Parameter space. Parameter choices; in case of hyperparameters with a list of choices, store the choices there.



**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

<code>classifiers.get_classifier(name, *args[, ...])</code>	Get an instance of a model from a string.
<code>classifiers.get_classifier_class(name)</code>	Get class of model from string.
<code>classifiers.list_classifiers()</code>	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.

## 24.3.4 asreview.models.query

Classes

<code>query.base.BaseQueryStrategy()</code>	Abstract class for query strategies.
<code>query.base.ProbaQueryStrategy()</code>	
<code>query.MaxQuery()</code>	Maximum query strategy ( <code>max</code> ).
<code>query.MixedQuery([strategy_1, strategy_2, ...])</code>	Mixed query strategy.
<code>query.MaxRandomQuery([mix_ratio, random_state])</code>	Mixed (95% Maximum and 5% Random) query strategy ( <code>max_random</code> ).
<code>query.MaxUncertaintyQuery([mix_ratio, ...])</code>	Mixed (95% Maximum and 5% Uncertainty) query strategy ( <code>max_uncertainty</code> ).
<code>query.UncertaintyQuery()</code>	Uncertainty query strategy ( <code>uncertainty</code> ).
<code>query.RandomQuery([random_state])</code>	Random query strategy ( <code>random</code> ).
<code>query.ClusterQuery([cluster_size, ...])</code>	Clustering query strategy ( <code>cluster</code> ).

## asreview.models.query.base.BaseQueryStrategy

**class** `asreview.models.query.base.BaseQueryStrategy`

Abstract class for query strategies.

### Attributes

<code>default_param</code>	Get the default parameters of the model.
<code>name</code>	
<code>param</code>	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])* Query new instances.

**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, **kwargs)`

Query new instances.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix to choose samples from.
- **classifier** (*SKLearnModel*) – Trained classifier to compute probabilities if they are necessary.
- **n\_instances** (*int*) – Number of instances to query.

**Returns**

(*numpy.ndarray*, *numpy.ndarray*) – The first is an array of shape (n\_instances,) containing the row indices of the new instances in query order. The second is an array of shape (n\_instances, n\_feature\_matrix\_columns), containing the feature vectors of the new instances.

**asreview.models.query.base.ProbaQueryStrategy****class** asreview.models.query.base.ProbaQueryStrategy**Attributes**

<i>default_param</i>	Get the default parameters of the model.
<i>name</i>	
<i>param</i>	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**

<i>full_hyper_space()</i>	
<i>hyper_space()</i>	
<i>query</i> (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*, *\*\*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**

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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**

<i>full_hyper_space()</i>	
<i>hyper_space()</i>	
<i>query</i> (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, **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

<i>default_param</i>	Get the default parameters of the model.
<i>name</i>	str(object=”) -> str str(bytes_or_buffer[, encoding[, errors]]) -> str
<i>param</i>	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

<code>full_hyper_space()</code>	
<code>hyper_space()</code>	
<code>query(X, classifier[, n_instances])</code>	Query new instances.

### `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, **kwargs)`

Query new instances.

#### Parameters

- **X** (`numpy.ndarray`) – Feature matrix to choose samples from.
- **classifier** (`SKLearnModel`) – Trained classifier to compute probabilities if they are necessary.
- **n\_instances** (`int`) – Number of instances to query.

#### Returns

(`numpy.ndarray`, `numpy.ndarray`) – The first is an array of shape (n\_instances,) containing the row indices of the new instances in query order. The second is an array of shape (n\_instances, n\_feature\_matrix\_columns), containing the feature vectors of the new instances.

### `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

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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

<i>full_hyper_space()</i>	
<i>hyper_space()</i>	
<i>query</i> (X, classifier[, n_instances])	Query new instances.

**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*, *\*\*kwargs*)

Query new instances.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix to choose samples from.
- **classifier** (*SKLearnModel*) – Trained classifier to compute probabilities if they are necessary.
- **n\_instances** (*int*) – Number of instances to query.

**Returns**

(*numpy.ndarray*, *numpy.ndarray*) – The first is an array of shape (n\_instances,) containing the row indices of the new instances in query order. The second is an array of shape (n\_instances, n\_feature\_matrix\_columns), containing the feature vectors of the new instances.

**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**

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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]) Query new instances.**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, **kwargs)`

Query new instances.

**Parameters**

- **X** (*numpy.ndarray*) – Feature matrix to choose samples from.
- **classifier** (*SKLearnModel*) – Trained classifier to compute probabilities if they are necessary.
- **n\_instances** (*int*) – Number of instances to query.

**Returns**

(*numpy.ndarray*, *numpy.ndarray*) – The first is an array of shape (n\_instances,) containing the row indices of the new instances in query order. The second is an array of shape (n\_instances, n\_feature\_matrix\_columns), containing the feature vectors of the new instances.

**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**

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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, **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**

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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

<code>full_hyper_space()</code>	
<code>hyper_space()</code>	
<code>query(X[, classifier, n_instances])</code>	Query new instances.

### `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, **kwargs)`

Query new instances.

#### Parameters

- **X** (`numpy.ndarray`) – Feature matrix to choose samples from.
- **classifier** (`SKLearnModel`) – Trained classifier to compute probabilities if they are necessary.
- **n\_instances** (`int`) – Number of instances to query.

#### Returns

(`numpy.ndarray`, `numpy.ndarray`) – The first is an array of shape (n\_instances,) containing the row indices of the new instances in query order. The second is an array of shape (n\_instances, n\_feature\_matrix\_columns), containing the feature vectors of the new instances.

### `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`, `RandomState`) – State/seed of the RNG.

## Attributes

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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

<i>full_hyper_space()</i>	
<i>hyper_space()</i>	
<i>query</i> (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, **kwargs)`

Query method for strategies which use class probabilities.

## Functions

<code>query.get_query_model(name, *args[, ...])</code>	Get an instance of the query strategy.
<code>query.get_query_class(name)</code>	Get class of query strategy from its name.
<code>query.list_query_strategies()</code>	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 also 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.

## 24.3.5 asreview.models.balance

Classes

<code>balance.base.BaseBalance()</code>	Abstract class for balance strategies.
<code>balance.SimpleBalance()</code>	No balance strategy ( <code>simple</code> ).
<code>balance.DoubleBalance([a, alpha, b, beta, ...])</code>	Double balance strategy ( <code>double</code> ).
<code>balance.TripleBalance([a, alpha, b, beta, ...])</code>	Triple balance strategy ( <code>triple</code> ).
<code>balance.UndersampleBalance([ratio, ran-dom_state])</code>	Undersampling balance strategy ( <code>undersample</code> ).

### asreview.models.balance.base.BaseBalance

**class** `asreview.models.balance.base.BaseBalance`

Abstract class for balance strategies.

**Attributes**

<code>default_param</code> <code>name</code>	Get the default parameters of the model.
<code>param</code>	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**

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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

<code>full_hyper_space()</code>	
<code>hyper_space()</code>	
<code>sample(X, y, train_idx)</code>	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

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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

<code>full_hyper_space()</code>	
<code>hyper_space()</code>	
<code>sample(X, y, train_idx)</code>	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 re-viewed.

#### 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

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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**

<i>full_hyper_space()</i>	
<i>hyper_space()</i>	
<i>sample(X, y, train_idx, shared)</i>	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

<i>default_param</i>	Get the default parameters of the model.
<i>label</i>	
<i>name</i>	
<i>param</i>	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

<code>full_hyper_space()</code>	
<code>hyper_space()</code>	
<code>sample(X, y, train_idx)</code>	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 re-viewed.

#### Returns

*numpy.ndarray, numpy.ndarray* – X\_train, y\_train: the resampled matrix, labels.

## Functions

<code>balance.get_balance_model(name, *args[, ...])</code>	Get an instance of a balance model from a string.
<code>balance.get_balance_class(name)</code>	Get class of balance model from string.
<code>balance.list_balance_strategies()</code>	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.

## 24.4 Projects and States

Load, interact, and extract information from project files and states (the “diary” of the review).

### 24.4.1 ASReviewProject

<code>ASReviewProject(project_path[, project_id])</code>	Project class for ASReview project files.
--	---

**asreview.ASReviewProject**

**class** `asreview.ASReviewProject(project_path, project_id=None)`

Project class for ASReview project files.

**Attributes**

*config*

*feature\_matrices*

*reviews*

**asreview.ASReviewProject.config****property** ASReviewProject.config**asreview.ASReviewProject.feature\_matrices****property** ASReviewProject.feature\_matrices**asreview.ASReviewProject.reviews****property** ASReviewProject.reviews**Methods**

<i>add_dataset</i> (file_name)	Add file path to the project file.
<i>add_feature_matrix</i> (feature_matrix, ...)	Add feature matrix to project file.
<i>add_review</i> (review_id[, start_time, status])	Add new review metadata.
<i>clean_tmp_files</i> ()	Clean temporary files in a project.
<i>create</i> (project_path[, project_id, ...])	Initialize the necessary files specific to the web app.
<i>delete_review</i> ([remove_folders])	
<i>export</i> (export_fp)	
<i>get_feature_matrix</i> (feature_extraction_method)	Get the feature matrix from the project file.
<i>load</i> (asreview_file, project_path[, safe_import])	
<i>mark_review_finished</i> ([review_id])	Mark a review in the project as finished.
<i>remove_dataset</i> ()	Remove dataset from project.
<i>remove_error</i> (status)	
<i>set_error</i> (err[, save_error_message])	
<i>update_config</i> (**kwargs)	Update project info
<i>update_review</i> ([review_id])	Update review metadata.

**asreview.ASReviewProject.add\_dataset****ASReviewProject.add\_dataset**(file\_name)

Add file path to the project file.

Add file to data subfolder and fill the pool of iteration 0.

**asreview.ASReviewProject.add\_feature\_matrix**

`ASReviewProject.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.ASReviewProject.add\_review**

`ASReviewProject.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.ASReviewProject.clean\_tmp\_files**

`ASReviewProject.clean_tmp_files()`

Clean temporary files in a project.

**Parameters**

- **project\_id** (*str*) – The id of the current project.

**asreview.ASReviewProject.create**

**classmethod** `ASReviewProject.create(project_path, project_id=None, project_mode='oracle', project_name=None, project_description=None, project_authors=None)`

Initialize the necessary files specific to the web app.

**asreview.ASReviewProject.delete\_review**

`ASReviewProject.delete_review(remove_folders=False)`

**asreview.ASReviewProject.export**

`ASReviewProject.export(export_fp)`

**asreview.ASReviewProject.get\_feature\_matrix**

`ASReviewProject.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.ASReviewProject.load**

**classmethod** `ASReviewProject.load(asreview_file, project_path, safe_import=False)`

**asreview.ASReviewProject.mark\_review\_finished**

`ASReviewProject.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.ASReviewProject.remove\_dataset**

`ASReviewProject.remove_dataset()`

Remove dataset from project.

**asreview.ASReviewProject.remove\_error**

`ASReviewProject.remove_error(status)`

**asreview.ASReviewProject.set\_error**

`ASReviewProject.set_error(err, save_error_message=True)`

**asreview.ASReviewProject.update\_config****ASReviewProject.update\_config(\*\*kwargs)**

Update project info

**asreview.ASReviewProject.update\_review****ASReviewProject.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.*) –

## 24.4.2 State

<code>open_state(asreview_obj[, review_id, read_only])</code>	Initialize a state class instance from a project folder.
<code>state.SQLiteState([read_only])</code>	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/ASReviewProject*) – Filepath to the (unzipped) project folder or ASReviewProject 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

<i>exist_new_labeled_records</i>	Return True if there are new labeled records.
<i>model_has_trained</i>	Return True if there is data of a trained model in the state.
<i>n_priors</i>	Number of records added as prior knowledge.
<i>n_records</i>	Number of records in the loop.
<i>n_records_labeled</i>	Number labeled records.
<i>settings</i>	Settings of the ASReview pipeline.
<i>version</i>	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**

<code>add_labeling_data(record_ids, labels[, ...])</code>	Add the data corresponding to a labeling action to the state file.
<code>add_last_probabilities(probabilities)</code>	Save the probabilities produced by the last classifier.
<code>add_last_ranking(ranked_record_ids, ...)</code>	Save the ranking of the last iteration of the model.
<code>add_note(note, record_id)</code>	Add a text note to save with a labeled record.
<code>add_record_table(record_ids)</code>	Add the record table to the state.
<code>close()</code>	Close the files opened by the state.

continues on next page

Table 1 – continued from previous page

<code>delete_record_labeling_data(record_id)</code>	Delete the labeling data for the given record id.
<code>get_balance_strategies([priors, pending])</code>	Get the balance strategies from the state.
<code>get_classifiers([priors, pending])</code>	Get the classifiers from the state.
<code>get_data_by_query_number(query[, columns])</code>	Get the data of a specific query from the results table.
<code>get_data_by_record_id(record_id[, columns])</code>	Get the data of a specific query from the results table.
<code>get_dataset([columns, priors, pending])</code>	Get a subset from the results table.
<code>get_decision_changes()</code>	Get the record ids for any decision changes.
<code>get_feature_extraction([priors, pending])</code>	Get the query strategies from the state.
<code>get_labeled()</code>	Get the labeled records in order of labeling.
<code>get_labeling_times([time_format, priors, ...])</code>	Get the time of labeling from the state.
<code>get_labels([priors, pending])</code>	Get the labels from the state.
<code>get_last_probabilities()</code>	Get the probabilities produced by the last classifier.
<code>get_last_ranking()</code>	Get the ranking from the state.
<code>get_order_of_labeling([priors, pending])</code>	Get full array of record id's in order that they were labeled.
<code>get_pending()</code>	Get the record_ids of the records pending a labeling decision.
<code>get_pool()</code>	Get the unlabeled, not-pending records in ranking order.
<code>get_pool_labeled_pending()</code>	Return the unlabeled pool, labeled and pending records.
<code>get_priors([columns])</code>	Get the record ids of the priors.
<code>get_query_strategies([priors, pending])</code>	Get the query strategies from the state.
<code>get_record_table()</code>	Get the record table of the state.
<code>get_training_sets([priors, pending])</code>	Get the training_sets from the state.
<code>is_empty()</code>	Check if state has no results.
<code>query_top_ranked(n)</code>	Get the top ranked records from the ranking table.
<code>to_dict()</code>	Convert state to dictionary.
<code>update_decision(record_id, label[, note])</code>	Change the label of an already labeled record.

### asreview.state.SQLiteState.add\_labeling\_data

`SQLiteState.add_labeling_data(record_ids, labels, notes=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.
- **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 probabilities for every record.

### 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.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=['record\_id'])**

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)`

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.

**24.4.3 Utils**

<code>project.get_project_path(folder_id)</code>	Get the project directory.
<code>project.project_from_id(f)</code>	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
<code>project.get_projects([project_paths])</code>	Get the ASReview projects at the given paths.
<code>project.is_project(project_path)</code>	
<code>project.is_v0_project(project_path)</code>	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[ASReviewProject]* – 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.

## 24.5 Readers and writers

This module contains the input and output functionality. You can install them as extensions.

<code>asreview.list_readers()</code>	List available dataset reader classes.
<code>asreview.list_writers()</code>	List available dataset writer classes.

### 24.5.1 asreview.list\_readers

asreview.list\_readers()

List available dataset reader classes.

**Returns**

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

### 24.5.2 asreview.list\_writers

asreview.list\_writers()

List available dataset writer classes.

**Returns**

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

<code>io.CSVReader()</code>	CVS file reader.
<code>io.CSVWriter()</code>	CSV file writer.
<code>io.ExcelReader()</code>	Excel file reader.
<code>io.ExcelWriter()</code>	Excel file writer.
<code>io.PaperRecord(record_id[, column_spec])</code>	A single record from a paper in a systematic review.
<code>io.RISReader()</code>	RIS file reader.
<code>io.RISWriter()</code>	RIS file writer.
<code>io.TSVWriter()</code>	TSV file writer.

### 24.5.3 asreview.io.CSVReader

**class** asreview.io.CSVReader

CVS file reader.

#### Attributes

`read_format`

`write_format`

#### asreview.io.CSVReader.read\_format

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

#### asreview.io.CSVReader.write\_format

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

#### Methods

`read_data(fp)`

Import dataset.

#### asreview.io.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.

## 24.5.4 asreview.io.CSVWriter

**class** asreview.io.CSVWriter

CSV file writer.

### Attributes

<i>label</i>
<i>name</i>
<i>write_format</i>

#### asreview.io.CSVWriter.label

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

#### asreview.io.CSVWriter.name

CSVWriter.name = 'csv'

#### asreview.io.CSVWriter.write\_format

CSVWriter.write\_format = '.csv'

### Methods

<i>write_data</i> (df, fp[, sep, labels, ranking])	Export dataset.
--	-----------------

#### asreview.io.CSVWriter.write\_data

**classmethod** CSVWriter.write\_data(df, fp, sep=',', labels=None, ranking=None)

Export dataset.

#### Parameters

- **df** (*pandas.DataFrame*) – Dataframe of all available record data.
- **fp** (*str*, *NoneType*) – Filepath or None for buffer.
- **sep** (*str*) – Separator of the file.
- **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 (internal) indices. Default ordering if ranking is None.

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

## 24.5.5 asreview.io.ExcelReader

**class** asreview.io.ExcelReader

Excel file reader.

**Attributes***read\_format**write\_format***asreview.io.ExcelReader.read\_format**

ExcelReader.read\_format = ['.xlsx']

**asreview.io.ExcelReader.write\_format**

ExcelReader.write\_format = ['.csv', '.tsv', '.xlsx']

**Methods***read\_data*(fp)

Import dataset.

**asreview.io.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.

## 24.5.6 asreview.io.ExcelWriter

**class** asreview.io.ExcelWriter

Excel file writer.

### Attributes

<i>label</i>
<i>name</i>
<i>write_format</i>

**asreview.io.ExcelWriter.label**

ExcelWriter.label = 'Excel'

**asreview.io.ExcelWriter.name**

ExcelWriter.name = 'xlsx'

**asreview.io.ExcelWriter.write\_format**

ExcelWriter.write\_format = '.xlsx'

### Methods

<i>write_data</i> (df, fp[, labels, ranking])	Export dataset.
---	-----------------

**asreview.io.ExcelWriter.write\_data**

**classmethod** ExcelWriter.write\_data(df, fp, labels=None, ranking=None)

Export dataset.

#### Parameters

- **df** (*pandas.DataFrame*) – Dataframe of all available record data.
- **fp** (*str*, *NoneType*) – Filepath or None for buffer.
- **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 (internal) indices. Default ordering if ranking is None.

#### Returns

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



## 24.5.7 asreview.io.PaperRecord

**class** asreview.io.PaperRecord(*record\_id*, *column\_spec*={}, *\*\*kwargs*)

A single record from a paper in a systematic review.

### Parameters

- **record\_id** (*int*) – Some identifier for this record.
- **title** (*str*) – Paper title.
- **abstract** (*str*) – Paper abstract.
- **authors** (*str*, *list*) – Authors of the paper.
- **notes** (*str*, *list*) – Notes of the paper.
- **keywords** (*str*, *list*) – Keywords of the paper.
- **label** (*int*) – Current label of the paper. No label is indicated by asreview.config.LABEL\_NA (== -1).
- **kwargs** (*dict*) – Any extra keyword arguments will be put in self.extra\_fields.

### Attributes

<i>body</i>	Return the abstract of the paper.
<i>heading</i>	Return the title of the paper.
<i>text</i>	Create a single string from title + abstract.

### asreview.io.PaperRecord.body

**property** PaperRecord.body

Return the abstract of the paper.

### asreview.io.PaperRecord.heading

**property** PaperRecord.heading

Return the title of the paper.

### asreview.io.PaperRecord.text

**property** PaperRecord.text

Create a single string from title + abstract.

### Returns

*str* – Concatenated string from title + abstract.

## Methods

### 24.5.8 asreview.io.RISReader

**class** asreview.io.RISReader

RIS file reader.

## Attributes

<code>read_format</code>
<code>write_format</code>

### asreview.io.RISReader.read\_format

RISReader.read\_format = ['.ris', '.txt']

### asreview.io.RISReader.write\_format

RISReader.write\_format = ['.csv', '.tsv', '.xlsx', '.ris']

## Methods

<code>read_data(fp)</code>	Import dataset.
----------------------------	-----------------

### asreview.io.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

*pandas.DataFrame* – Dataframe with entries.

#### Raises

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

### 24.5.9 asreview.io.RISWriter

**class** asreview.io.RISWriter

RIS file writer.

#### Attributes

<i>caution</i>
<i>label</i>
<i>name</i>
<i>write_format</i>

#### asreview.io.RISWriter.caution

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

#### asreview.io.RISWriter.label

`RISWriter.label = 'RIS'`

#### asreview.io.RISWriter.name

`RISWriter.name = 'ris'`

#### asreview.io.RISWriter.write\_format

`RISWriter.write_format = '.ris'`

#### Methods

<i>write_data</i> (df, fp[, labels, ranking])	Export dataset.
---	-----------------

**asreview.io.RISWriter.write\_data****classmethod** `RISWriter.write_data(df, fp, labels=None, ranking=None)`

Export dataset.

**Parameters**

- **df** (*pandas.DataFrame*) – Dataframe of all available record data.
- **fp** (*str*, *pathlib.Path*) – File path to the RIS file, if exists.
- **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 (internal) indices. Default ordering if ranking is None.

**Returns***RIS file* – Dataframe of all available record data.**24.5.10 asreview.io.TSVWriter****class** `asreview.io.TSVWriter`

TSV file writer.

**Attributes**

<i>label</i>
<i>name</i>
<i>write_format</i>

**asreview.io.TSVWriter.label**`TSVWriter.label = 'TSV (UTF-8)'`**asreview.io.TSVWriter.name**`TSVWriter.name = 'tsv'`

**asreview.io.TSVWriter.write\_format**

```
TSVWriter.write_format = '.tsv'
```

**Methods**

<code>write_data(df, fp[, sep, labels, ranking])</code>	Export dataset.
---	-----------------

**asreview.io.TSVWriter.write\_data**

```
classmethod TSVWriter.write_data(df, fp, sep='\t', labels=None, ranking=None)
```

Export dataset.

**Parameters**

- **df** (*pandas.DataFrame*) – Dataframe of all available record data.
- **fp** (*str*, *NoneType*) – Filepath or None for buffer.
- **sep** (*str*) – Separator of the file.
- **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 (internal) indices. Default ordering if ranking is None.

**Returns**

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

## 24.6 Misc

**Classes**

<code>asreview.settings.ASReviewSettings(model, ...)</code>	Object to store the configuration of a review session.
---	--

### 24.6.1 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={},
                                         query_param={}, balance_param={}, feature_param={},
                                         data_fp=None, n_queries=None, abstract_only=False,
                                         mode=None, n_papers=None, data_name=None)
```

Object to store the configuration of a review session.

The main difference being that it type checks (some) of its contents.

## Methods

<code>from_file(config_file)</code>	Fill the contents of settings by reading a config file.
<code>to_dict()</code>	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

<code>search.fuzzy_find(as_data, keywords[, ...])</code>	Find a record using keywords.
<code>asreview_path()</code>	Get the location where projects are stored.
<code>get_data_home([data_home])</code>	Return the path of the ASR data dir.

## 24.6.2 `asreview.search.fuzzy_find`

`asreview.search.fuzzy_find(as_data, keywords, threshold=60, max_return=10, exclude=None, by_index=True)`

Find a record using keywords.

It looks for keywords in the title/authors/keywords (for as much is available). Using the difflib package it creates a ranking based on token set matching.

#### Parameters

- **as\_data** (`asreview.data.ASReviewData`) – 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.
- **by\_index** (*bool*) – If True, use internal indexing. If False, use record ids for indexing.

#### Returns

*list* – Sorted list of indexes that match best the keywords.

### 24.6.3 asreview.asreview\_path

`asreview.asreview_path()`

Get the location where projects are stored.

Overwrite this location by specifying the ASREVIEW\_PATH environment variable.

### 24.6.4 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.

## 24.7 Entry points

Entry points for ASReview LAB.

<code>entry_points.BaseEntryPoint()</code>	Base class for defining entry points.
<code>entry_points.AlgorithmsEntryPoint()</code>	Entry point to list available algorithms in ASReview LAB.
<code>entry_points.LABEntryPoint()</code>	Entry point to start the ASReview LAB webapp.
<code>entry_points.SimulateEntryPoint()</code>	Entry point for simulation with ASReview LAB.
<code>entry_points.StateInspectEntryPoint()</code>	Entry point to inspect ASReview LAB review progress.

### 24.7.1 asreview.entry\_points.BaseEntryPoint

**class** `asreview.entry_points.BaseEntryPoint`

Base class for defining entry points.

#### Methods

<code>execute(argv)</code>	Perform the functionality of the entry point.
----------------------------	---

**asreview.entry\_points.BaseEntryPoint.execute****abstract classmethod** BaseEntryPoint.execute(*argv*)

Perform the functionality of the entry point.

**Parameters****argv** (*list*) – Argument list, with the entry point and program removed. For example, if *asreview plot X* is executed, then `argv == ['X']`.**24.7.2 asreview.entry\_points.AlgorithmsEntryPoint****class** asreview.entry\_points.AlgorithmsEntryPoint

Entry point to list available algorithms in ASReview LAB.

**Attributes**

<i>description</i>
--------------------

**asreview.entry\_points.AlgorithmsEntryPoint.description**`AlgorithmsEntryPoint.description = 'Available active learning algorithms for ASReview.'`**Methods**

<i>execute</i> ( <i>argv</i> )	Perform the functionality of the entry point.
--------------------------------	---

**asreview.entry\_points.AlgorithmsEntryPoint.execute**`AlgorithmsEntryPoint.execute(argv)`

Perform the functionality of the entry point.

**Parameters****argv** (*list*) – Argument list, with the entry point and program removed. For example, if *asreview plot X* is executed, then `argv == ['X']`.



### 24.7.3 asreview.entry\_points.LABEntryPoint

**class** asreview.entry\_points.LABEntryPoint

Entry point to start the ASReview LAB webapp.

#### Methods

<code>execute(argv)</code>	Perform the functionality of the entry point.
----------------------------	---

#### asreview.entry\_points.LABEntryPoint.execute

LABEntryPoint.**execute**(*argv*)

Perform the functionality of the entry point.

##### Parameters

**argv** (*list*) – Argument list, with the entry point and program removed. For example, if *asreview plot X* is executed, then `argv == ['X']`.

### 24.7.4 asreview.entry\_points.SimulateEntryPoint

**class** asreview.entry\_points.SimulateEntryPoint

Entry point for simulation with ASReview LAB.

#### Methods

<code>execute(argv)</code>	Perform the functionality of the entry point.
----------------------------	---

#### asreview.entry\_points.SimulateEntryPoint.execute

SimulateEntryPoint.**execute**(*argv*)

Perform the functionality of the entry point.

##### Parameters

**argv** (*list*) – Argument list, with the entry point and program removed. For example, if *asreview plot X* is executed, then `argv == ['X']`.

### 24.7.5 asreview.entry\_points.StateInspectEntryPoint

**class** asreview.entry\_points.StateInspectEntryPoint

Entry point to inspect ASReview LAB review progress.

## Methods

<code>execute(argv)</code>	Perform the functionality of the entry point.
----------------------------	---

### `asreview.entry_points.StateInspectEntryPoint.execute`

`StateInspectEntryPoint.execute(argv)`

Perform the functionality of the entry point.

#### Parameters

**argv** (*list*) – Argument list, with the entry point and program removed. For example, if *asreview plot X* is executed, then `argv == ['X']`.

## INDICES AND TABLES

- `genindex`
- `modindex`



## PYTHON MODULE INDEX

### a

- `asreview`, 89
- `asreview.data`, 89
- `asreview.datasets`, 98
- `asreview.entry_points`, 195
- `asreview.io`, 185
- `asreview.models`, 107
  - `asreview.models.balance`, 158
  - `asreview.models.classifiers`, 123
  - `asreview.models.feature_extraction`, 108
  - `asreview.models.query`, 142
- `asreview.project`, 168
- `asreview.review`, 104
- `asreview.state`, 172



## Symbols

--balance\_strategy  
     command line option, 64  
 --certfile  
     asreview-lab command line option, 18  
 --clean-all-projects  
     asreview-lab command line option, 18  
 --clean-project  
     asreview-lab command line option, 18  
 --embedding  
     asreview-lab command line option, 18  
     command line option, 64  
 --enable-auth  
     asreview-lab command line option, 17  
 --feature\_extraction  
     command line option, 64  
 --flask-configfile  
     asreview-lab command line option, 17  
 --help  
     asreview-lab command line option, 17  
 --init\_seed  
     command line option, 65  
 --ip  
     asreview-lab command line option, 17  
 --keyfile  
     asreview-lab command line option, 18  
 --model  
     command line option, 64  
 --n\_instances  
     command line option, 65  
 --n\_prior\_excluded  
     command line option, 64  
 --n\_prior\_included  
     command line option, 64  
 --no-browser  
     asreview-lab command line option, 17  
 --port  
     asreview-lab command line option, 17  
 --port-retries  
     asreview-lab command line option, 17  
 --prior\_idx  
     command line option, 64

--query\_strategy  
     command line option, 64  
 --salt  
     asreview-lab command line option, 17  
 --secret-key  
     asreview-lab command line option, 17  
 --seed  
     asreview-lab command line option, 18  
     command line option, 64  
 --state\_file  
     command line option, 65  
 --stop\_if  
     command line option, 65  
 -b  
     command line option, 64  
 -e  
     command line option, 64  
 -h  
     asreview-lab command line option, 17  
 -m  
     command line option, 64  
 -q  
     command line option, 64  
 -s  
     command line option, 65

## A

abstract (*asreview.ASReviewData* property), 91  
 abstract\_length() (in module *asreview.data.statistics*), 96  
 Active learning model, 5  
 add\_dataset() (*asreview.ASReviewProject* method), 169  
 add\_feature\_matrix() (*asreview.ASReviewProject* method), 170  
 add\_labeling\_data() (*asreview.state.SQLiteState* method), 175  
 add\_last\_probabilities() (*asreview.state.SQLiteState* method), 176  
 add\_last\_ranking() (*asreview.state.SQLiteState* method), 176  
 add\_note() (*asreview.state.SQLiteState* method), 176

`add_record_table()` (*asreview.state.SQLiteState* method), 176

`add_review()` (*asreview.ASReviewProject* method), 170

`AlgorithmsEntryPoint` (class in *asreview.entry\_points*), 196

`append()` (*asreview.datasets.BaseDataGroup* method), 102

`append()` (*asreview.datasets.NaturePublicationDataGroup* method), 100

`append()` (*asreview.datasets.SynergyDataGroup* method), 99

`ASReview`, 5

`asreview`

- module, 89

`ASReview CLI`, 5

`asreview.data`

- module, 89

`asreview.datasets`

- module, 98

`asreview.entry_points`

- module, 195

`asreview.io`

- module, 185

`asreview.models`

- module, 107

`asreview.models.balance`

- module, 158

`asreview.models.classifiers`

- module, 123

`asreview.models.feature_extraction`

- module, 108

`asreview.models.query`

- module, 142

`asreview.project`

- module, 168

`asreview.review`

- module, 104

`asreview.state`

- module, 172

`ASREVIEW_PATH`

- `asreview-lab` command line option, 18

`asreview_path()` (*in module asreview*), 195

`asreview-lab` command line option

- `--certfile`, 18
- `--clean-all-projects`, 18
- `--clean-project`, 18
- `--embedding`, 18
- `--enable-auth`, 17
- `--flask-configfile`, 17
- `--help`, 17
- `--ip`, 17
- `--keyfile`, 18
- `--no-browser`, 17
- `--port`, 17
- `--port-retries`, 17
- `--salt`, 17
- `--secret-key`, 17
- `--seed`, 18
- `-h`, 17
- `ASREVIEW_PATH`, 18
- `ASReviewData` (class in *asreview*), 89
- `ASReviewProject` (class in *asreview*), 168
- `ASReviewSettings` (class in *asreview.settings*), 193
- `authors` (*asreview.ASReviewData* property), 91

## B

`BaseBalance` (class in *asreview.models.balance.base*), 158

`BaseDataGroup` (class in *asreview.datasets*), 101

`BaseDataSet` (class in *asreview.datasets*), 101

`BaseEntryPoint` (class in *asreview.entry\_points*), 195

`BaseFeatureExtraction` (class in *asreview.models.feature\_extraction.base*), 108

`BaseModel` (class in *asreview.models.base*), 107

`BaseQueryStrategy` (class in *asreview.models.query.base*), 142

`BaseReview` (class in *asreview.review*), 104

`BaseTrainClassifier` (class in *asreview.models.classifiers.base*), 124

`bodies` (*asreview.ASReviewData* property), 91

`body` (*asreview.io.PaperRecord* property), 189

## C

`caution` (*asreview.io.RISWriter* attribute), 191

`clean_tmp_files()` (*asreview.ASReviewProject* method), 170

`close()` (*asreview.state.SQLiteState* method), 177

`ClusterQuery` (class in *asreview.models.query*), 155

command line option

- `--balance_strategy`, 64
- `--embedding`, 64
- `--feature_extraction`, 64
- `--init_seed`, 65
- `--model`, 64
- `--n_instances`, 65
- `--n_prior_excluded`, 64
- `--n_prior_included`, 64
- `--prior_idx`, 64
- `--query_strategy`, 64
- `--seed`, 64
- `--state_file`, 65
- `--stop_if`, 65
- `-b`, 64
- `-e`, 64
- `-m`, 64
- `-q`, 64
- `-s`, 65
- dataset, 64



`config` (*asreview.ASReviewProject* property), 169  
`create()` (*asreview.ASReviewProject* class method), 170  
`CSVReader` (class in *asreview.io*), 185  
`CSVWriter` (class in *asreview.io*), 186

## D

`Data`, 5

`Dataset`, 5

`dataset`

command line option, 64

`DatasetManager` (class in *asreview.datasets*), 103

`default_param` (*asreview.view.models.balance.base.BaseBalance* property), 158

`default_param` (*asreview.view.models.balance.DoubleBalance* property), 162

`default_param` (*asreview.view.models.balance.SimpleBalance* property), 160

`default_param` (*asreview.view.models.balance.TripleBalance* property), 164

`default_param` (*asreview.view.models.balance.UndersampleBalance* property), 166

`default_param` (*asreview.models.base.BaseModel* property), 107

`default_param` (*asreview.view.models.classifiers.base.BaseTrainClassifier* property), 124

`default_param` (*asreview.view.models.classifiers.LogisticClassifier* property), 132

`default_param` (*asreview.view.models.classifiers.LSTMBaseClassifier* property), 134

`default_param` (*asreview.view.models.classifiers.LSTMPoolClassifier* property), 137

`default_param` (*asreview.view.models.classifiers.NaiveBayesClassifier* property), 126

`default_param` (*asreview.view.models.classifiers.NN2LayerClassifier* property), 139

`default_param` (*asreview.view.models.classifiers.RandomForestClassifier* property), 128

`default_param` (*asreview.view.models.classifiers.SVMClassifier* property), 130

`default_param` (*asreview.view.models.feature\_extraction.base.BaseFeatureExtraction*

property), 109

`default_param` (*asreview.view.models.feature\_extraction.Doc2Vec* property), 113

`default_param` (*asreview.view.models.feature\_extraction.EmbeddingIdf* property), 116

`default_param` (*asreview.view.models.feature\_extraction.EmbeddingLSTM* property), 118

`default_param` (*asreview.view.models.feature\_extraction.SBERT* property), 121

`default_param` (*asreview.view.models.feature\_extraction.Tfidf* property), 111

`default_param` (*asreview.view.models.query.base.BaseQueryStrategy* property), 142

`default_param` (*asreview.view.models.query.base.ProbaQueryStrategy* property), 144

`default_param` (*asreview.models.query.ClusterQuery* property), 156

`default_param` (*asreview.models.query.MaxQuery* property), 145

`default_param` (*asreview.view.models.query.MaxRandomQuery* property), 149

`default_param` (*asreview.view.models.query.MaxUncertaintyQuery* property), 151

`default_param` (*asreview.models.query.MixedQuery* property), 147

`default_param` (*asreview.models.query.RandomQuery* property), 154

`default_param` (*asreview.view.models.query.UncertaintyQuery* property), 152

`delete_record_labeling_data()` (*asreview.view.state.SQLiteState* method), 177

`delete_review()` (*asreview.ASReviewProject* method), 170

`description` (*asreview.datasets.BaseDataGroup* property), 102

`description` (*asreview.datasets.NaturePublicationDataGroup* attribute), 99

`description` (*asreview.datasets.SynergyDataGroup* attribute), 98

`description` (*asreview.entry\_points.AlgorithmsEntryPoint* attribute), 196

`Doc2Vec` (class in *asreview.models.feature\_extraction*), 112

`doi` (*asreview.ASReviewData* property), 91

DoubleBalance (*class in asreview.models.balance*), 161  
 drop\_duplicates() (*asreview.ASReviewData method*), 93  
 duplicated() (*asreview.ASReviewData method*), 93

## E

ELAS, 5

EmbeddingIdf (*class in asreview.models.feature\_extraction*), 115

EmbeddingLSTM (*class in asreview.models.feature\_extraction*), 117

ExcelReader (*class in asreview.io*), 187

ExcelWriter (*class in asreview.io*), 188

execute() (*asreview.entry\_points.AlgorithmsEntryPoint method*), 196

execute() (*asreview.entry\_points.BaseEntryPoint class method*), 196

execute() (*asreview.entry\_points.LABEntryPoint method*), 197

execute() (*asreview.entry\_points.SimulateEntryPoint method*), 197

execute() (*asreview.entry\_points.StateInspectEntryPoint method*), 198

exist\_new\_labeled\_records (*asreview.state.SQLiteState property*), 173

Export, 5

export() (*asreview.ASReviewProject method*), 171

Extension, 5

## F

feature\_matrices (*asreview.ASReviewProject property*), 169

filename (*asreview.datasets.BaseDataSet property*), 101

final\_included (*asreview.ASReviewData property*), 91

find() (*asreview.datasets.BaseDataGroup method*), 102

find() (*asreview.datasets.DatasetManager method*), 103

find() (*asreview.datasets.NaturePublicationDataGroup method*), 100

find() (*asreview.datasets.SynergyDataGroup method*), 99

fit() (*asreview.models.classifiers.base.BaseTrainClassifier method*), 125

fit() (*asreview.models.classifiers.LogisticClassifier method*), 133

fit() (*asreview.models.classifiers.LSTMBaseClassifier method*), 135

fit() (*asreview.models.classifiers.LSTMPoolClassifier method*), 138

fit() (*asreview.models.classifiers.NaiveBayesClassifier method*), 127

fit() (*asreview.models.classifiers.NN2LayerClassifier method*), 140

fit() (*asreview.models.classifiers.RandomForestClassifier method*), 129

fit() (*asreview.models.classifiers.SVMClassifier method*), 131

fit() (*asreview.models.feature\_extraction.base.BaseFeatureExtraction method*), 109

fit() (*asreview.models.feature\_extraction.Doc2Vec method*), 114

fit() (*asreview.models.feature\_extraction.EmbeddingIdf method*), 116

fit() (*asreview.models.feature\_extraction.EmbeddingLSTM method*), 119

fit() (*asreview.models.feature\_extraction.SBERT method*), 122

fit() (*asreview.models.feature\_extraction.Tfidf method*), 112

fit\_transform() (*asreview.models.feature\_extraction.base.BaseFeatureExtraction method*), 110

fit\_transform() (*asreview.models.feature\_extraction.Doc2Vec method*), 114

fit\_transform() (*asreview.models.feature\_extraction.EmbeddingIdf method*), 117

fit\_transform() (*asreview.models.feature\_extraction.EmbeddingLSTM method*), 119

fit\_transform() (*asreview.models.feature\_extraction.SBERT method*), 122

fit\_transform() (*asreview.models.feature\_extraction.Tfidf method*), 112

from\_extension() (*asreview.ASReviewData class method*), 93

from\_file() (*asreview.ASReviewData class method*), 93

from\_file() (*asreview.settings.ASReviewSettings method*), 194

full\_hyper\_space() (*asreview.models.balance.base.BaseBalance method*), 159

full\_hyper\_space() (*asreview.models.balance.DoubleBalance method*), 163

full\_hyper\_space() (*asreview.models.balance.SimpleBalance method*), 161

full\_hyper\_space() (*asreview.models.balance.TripleBalance method*), 165

full\_hyper\_space() (*asreview.models.balance.UndersampleBalance method*), 165

- method), 167
- full\_hyper\_space() (asreview.models.base.BaseModel method), 108
- full\_hyper\_space() (asreview.models.classifiers.base.BaseTrainClassifier method), 125
- full\_hyper\_space() (asreview.models.classifiers.LogisticClassifier method), 133
- full\_hyper\_space() (asreview.models.classifiers.LSTMBaseClassifier method), 135
- full\_hyper\_space() (asreview.models.classifiers.LSTMPoolClassifier method), 138
- full\_hyper\_space() (asreview.models.classifiers.NaiveBayesClassifier method), 127
- full\_hyper\_space() (asreview.models.classifiers.NN2LayerClassifier method), 140
- full\_hyper\_space() (asreview.models.classifiers.RandomForestClassifier method), 129
- full\_hyper\_space() (asreview.models.classifiers.SVMClassifier method), 131
- full\_hyper\_space() (asreview.models.feature\_extraction.base.BaseFeatureExtraction method), 110
- full\_hyper\_space() (asreview.models.feature\_extraction.Doc2Vec method), 115
- full\_hyper\_space() (asreview.models.feature\_extraction.EmbeddingIdf method), 117
- full\_hyper\_space() (asreview.models.feature\_extraction.EmbeddingLSTM method), 119
- full\_hyper\_space() (asreview.models.feature\_extraction.SBERT method), 122
- full\_hyper\_space() (asreview.models.feature\_extraction.Tfidf method), 112
- full\_hyper\_space() (asreview.models.query.base.BaseQueryStrategy method), 143
- full\_hyper\_space() (asreview.models.query.base.ProbaQueryStrategy method), 145
- full\_hyper\_space() (asreview.models.query.ClusterQuery method), 157
- full\_hyper\_space() (asreview.view.models.query.MaxQuery method), 146
- full\_hyper\_space() (asreview.view.models.query.MaxRandomQuery method), 150
- full\_hyper\_space() (asreview.view.models.query.MaxUncertaintyQuery method), 151
- full\_hyper\_space() (asreview.view.models.query.MixedQuery method), 148
- full\_hyper\_space() (asreview.view.models.query.RandomQuery method), 155
- full\_hyper\_space() (asreview.view.models.query.UncertaintyQuery method), 153
- fuzzy\_find() (in module asreview.search), 194
- ## G
- get() (asreview.ASReviewData method), 94
- get\_balance\_class() (in module asreview.models.balance), 168
- get\_balance\_model() (in module asreview.models.balance), 167
- get\_balance\_strategies() (asreview.state.SQLiteState method), 177
- get\_classifier() (in module asreview.models.classifiers), 141
- get\_classifier\_class() (in module asreview.models.classifiers), 141
- get\_classifiers() (asreview.state.SQLiteState method), 177
- get\_data\_by\_query\_number() (asreview.state.SQLiteState method), 177
- get\_data\_by\_record\_id() (asreview.state.SQLiteState method), 178
- get\_data\_home() (in module asreview), 195
- get\_dataset() (asreview.state.SQLiteState method), 178
- get\_decision\_changes() (asreview.state.SQLiteState method), 178
- get\_embedding\_matrix() (asreview.models.feature\_extraction.EmbeddingLSTM method), 119
- get\_feature\_class() (in module asreview.models.feature\_extraction), 123
- get\_feature\_extraction() (asreview.state.SQLiteState method), 179
- get\_feature\_matrix() (asreview.ASReviewProject method), 171
- get\_feature\_model() (in module asreview.models.feature\_extraction), 123

`get_labeled()` (*asreview.state.SQLiteState* method), 179  
`get_labeling_times()` (*asreview.state.SQLiteState* method), 179  
`get_labels()` (*asreview.state.SQLiteState* method), 179  
`get_last_probabilities()` (*asreview.state.SQLiteState* method), 180  
`get_last_ranking()` (*asreview.state.SQLiteState* method), 180  
`get_order_of_labeling()` (*asreview.state.SQLiteState* method), 180  
`get_pending()` (*asreview.state.SQLiteState* method), 180  
`get_pool()` (*asreview.state.SQLiteState* method), 181  
`get_pool_labeled_pending()` (*asreview.state.SQLiteState* method), 181  
`get_priors()` (*asreview.state.SQLiteState* method), 181  
`get_project_path()` (in module *asreview.project*), 183  
`get_projects()` (in module *asreview.project*), 184  
`get_query_class()` (in module *asreview.models.query*), 157  
`get_query_model()` (in module *asreview.models.query*), 157  
`get_query_strategies()` (*asreview.state.SQLiteState* method), 181  
`get_record_table()` (*asreview.state.SQLiteState* method), 182  
`get_training_sets()` (*asreview.state.SQLiteState* method), 182  
`group_id` (*asreview.datasets.BaseDataGroup* property), 102  
`group_id` (*asreview.datasets.NaturePublicationDataGroup* attribute), 100  
`group_id` (*asreview.datasets.SynergyDataGroup* attribute), 98  
`groups` (*asreview.datasets.DatasetManager* property), 103

## H

`hash()` (*asreview.ASReviewData* method), 94  
`heading` (*asreview.io.PaperRecord* property), 189  
`headings` (*asreview.ASReviewData* property), 91  
`hyper_space()` (*asreview.models.balance.base.BaseBalance* method), 159  
`hyper_space()` (*asreview.models.balance.DoubleBalance* method), 163  
`hyper_space()` (*asreview.models.balance.SimpleBalance* method), 161  
`hyper_space()` (*asreview.models.balance.TripleBalance* method), 165  
`hyper_space()` (*asreview.models.balance.UndersampleBalance* method), 167  
`hyper_space()` (*asreview.models.base.BaseModel* method), 108  
`hyper_space()` (*asreview.models.classifiers.base.BaseTrainClassifier* method), 125  
`hyper_space()` (*asreview.models.classifiers.LogisticClassifier* method), 133  
`hyper_space()` (*asreview.models.classifiers.LSTMBaseClassifier* method), 136  
`hyper_space()` (*asreview.models.classifiers.LSTMPoolClassifier* method), 138  
`hyper_space()` (*asreview.models.classifiers.NaiveBayesClassifier* method), 127  
`hyper_space()` (*asreview.models.classifiers.NN2LayerClassifier* method), 141  
`hyper_space()` (*asreview.models.classifiers.RandomForestClassifier* method), 129  
`hyper_space()` (*asreview.models.classifiers.SVMClassifier* method), 131  
`hyper_space()` (*asreview.models.feature\_extraction.base.BaseFeatureExtraction* method), 110  
`hyper_space()` (*asreview.models.feature\_extraction.Doc2Vec* method), 115  
`hyper_space()` (*asreview.models.feature\_extraction.EmbeddingIdf* method), 117  
`hyper_space()` (*asreview.models.feature\_extraction.EmbeddingLSTM* method), 120  
`hyper_space()` (*asreview.models.feature\_extraction.SBERT* method), 122  
`hyper_space()` (*asreview.models.feature\_extraction.Tfidf* method), 112  
`hyper_space()` (*asreview.models.query.base.BaseQueryStrategy* method), 143  
`hyper_space()` (*asreview.models.query.base.ProbaQueryStrategy* method), 145  
`hyper_space()` (*asreview.models.query.ClusterQuery*

- method), 157
- `hyper_space()` (*asreview.models.query.MaxQuery* method), 146
- `hyper_space()` (*asreview.models.query.MaxRandomQuery* method), 150
- `hyper_space()` (*asreview.models.query.MaxUncertaintyQuery* method), 151
- `hyper_space()` (*asreview.models.query.MixedQuery* method), 148
- `hyper_space()` (*asreview.models.query.RandomQuery* method), 155
- `hyper_space()` (*asreview.models.query.UncertaintyQuery* method), 153
- ## I
- `Import`, 5
- `included` (*asreview.ASReviewData* property), 91
- `is_empty()` (*asreview.state.SQLiteState* method), 182
- `is_project()` (in module *asreview.project*), 184
- `is_v0_project()` (in module *asreview.project*), 184

## K

`keywords` (*asreview.ASReviewData* property), 91

## L

`label` (*asreview.io.CSVWriter* attribute), 186

`label` (*asreview.io.ExcelWriter* attribute), 188

`label` (*asreview.io.RISWriter* attribute), 191

`label` (*asreview.io.TSVWriter* attribute), 192

`label` (*asreview.models.balance.DoubleBalance* attribute), 162

`label` (*asreview.models.balance.SimpleBalance* attribute), 160

`label` (*asreview.models.balance.TripleBalance* attribute), 164

`label` (*asreview.models.balance.UndersampleBalance* attribute), 166

`label` (*asreview.models.classifiers.LogisticClassifier* attribute), 132

`label` (*asreview.models.classifiers.LSTMBaseClassifier* attribute), 134

`label` (*asreview.models.classifiers.LSTMPoolClassifier* attribute), 137

`label` (*asreview.models.classifiers.NaiveBayesClassifier* attribute), 126

`label` (*asreview.models.classifiers.NN2LayerClassifier* attribute), 139

`label` (*asreview.models.classifiers.RandomForestClassifier* attribute), 128

`label` (*asreview.models.classifiers.SVMClassifier* attribute), 130

`label` (*asreview.models.feature\_extraction.Doc2Vec* attribute), 113

`label` (*asreview.models.feature\_extraction.EmbeddingIdf* attribute), 116

`label` (*asreview.models.feature\_extraction.EmbeddingLSTM* attribute), 118

`label` (*asreview.models.feature\_extraction.SBERT* attribute), 121

`label` (*asreview.models.feature\_extraction.Tfidf* attribute), 111

`label` (*asreview.models.query.ClusterQuery* attribute), 156

`label` (*asreview.models.query.MaxQuery* attribute), 145

`label` (*asreview.models.query.MaxRandomQuery* attribute), 149

`label` (*asreview.models.query.MaxUncertaintyQuery* attribute), 151

`label` (*asreview.models.query.RandomQuery* attribute), 154

`label` (*asreview.models.query.UncertaintyQuery* attribute), 153

`labels` (*asreview.ASReviewData* property), 91

`LABEntryPoint` (class in *asreview.entry\_points*), 197

`list()` (*asreview.datasets.DatasetManager* method), 103

`list_balance_strategies()` (in module *asreview.models.balance*), 168

`list_classifiers()` (in module *asreview.models.classifiers*), 142

`list_feature_extraction()` (in module *asreview.models.feature\_extraction*), 123

`list_query_strategies()` (in module *asreview.models.query*), 158

`list_readers()` (in module *asreview*), 184

`list_writers()` (in module *asreview*), 184

`load()` (*asreview.ASReviewProject* class method), 171

`load_data()` (in module *asreview*), 89

`LogisticClassifier` (class in *asreview.models.classifiers*), 131

`LSTMBaseClassifier` (class in *asreview.models.classifiers*), 133

`LSTMPoolClassifier` (class in *asreview.models.classifiers*), 136

## M

`mark_review_finished()` (*asreview.ASReviewProject* method), 171

`MaxQuery` (class in *asreview.models.query*), 145

`MaxRandomQuery` (class in *asreview.models.query*), 148

`MaxUncertaintyQuery` (class in *asreview.models.query*), 150

`MixedQuery` (class in *asreview.models.query*), 146

`Model` configuration, 5



`model_has_trained` (*asreview.state.SQLiteState* property), 173

`module`

- `asreview`, 89
- `asreview.data`, 89
- `asreview.datasets`, 98
- `asreview.entry_points`, 195
- `asreview.io`, 185
- `asreview.models`, 107
- `asreview.models.balance`, 158
- `asreview.models.classifiers`, 123
- `asreview.models.feature_extraction`, 108
- `asreview.models.query`, 142
- `asreview.project`, 168
- `asreview.review`, 104
- `asreview.state`, 172

## N

`n_duplicates()` (in module *asreview.data.statistics*), 96

`n_irrelevant()` (in module *asreview.data.statistics*), 96

`n_keywords()` (in module *asreview.data.statistics*), 96

`n_missing_abstract()` (in module *asreview.data.statistics*), 97

`n_missing_title()` (in module *asreview.data.statistics*), 97

`n_priors` (*asreview.state.SQLiteState* property), 173

`n_records` (*asreview.state.SQLiteState* property), 174

`n_records()` (in module *asreview.data.statistics*), 97

`n_records_labeled` (*asreview.state.SQLiteState* property), 174

`n_relevant()` (in module *asreview.data.statistics*), 97

`n_unlabeled()` (in module *asreview.data.statistics*), 97

`NaiveBayesClassifier` (class in *asreview.models.classifiers*), 125

`name` (*asreview.io.CSVWriter* attribute), 186

`name` (*asreview.io.ExcelWriter* attribute), 188

`name` (*asreview.io.RISWriter* attribute), 191

`name` (*asreview.io.TSVWriter* attribute), 192

`name` (*asreview.models.balance.base.BaseBalance* attribute), 159

`name` (*asreview.models.balance.DoubleBalance* attribute), 162

`name` (*asreview.models.balance.SimpleBalance* attribute), 160

`name` (*asreview.models.balance.TripleBalance* attribute), 164

`name` (*asreview.models.balance.UndersampleBalance* attribute), 166

`name` (*asreview.models.base.BaseModel* attribute), 107

`name` (*asreview.models.classifiers.base.BaseTrainClassifier* attribute), 124

`name` (*asreview.models.classifiers.LogisticClassifier* attribute), 132

`name` (*asreview.models.classifiers.LSTMBaseClassifier* attribute), 135

`name` (*asreview.models.classifiers.LSTMPoolClassifier* attribute), 137

`name` (*asreview.models.classifiers.NaiveBayesClassifier* attribute), 126

`name` (*asreview.models.classifiers.NN2LayerClassifier* attribute), 140

`name` (*asreview.models.classifiers.RandomForestClassifier* attribute), 128

`name` (*asreview.models.classifiers.SVMClassifier* attribute), 130

`name` (*asreview.models.feature\_extraction.base.BaseFeatureExtraction* attribute), 109

`name` (*asreview.models.feature\_extraction.Doc2Vec* attribute), 114

`name` (*asreview.models.feature\_extraction.EmbeddingIdf* attribute), 116

`name` (*asreview.models.feature\_extraction.EmbeddingLSTM* attribute), 118

`name` (*asreview.models.feature\_extraction.SBERT* attribute), 121

`name` (*asreview.models.feature\_extraction.Tfidf* attribute), 111

`name` (*asreview.models.query.base.BaseQueryStrategy* attribute), 143

`name` (*asreview.models.query.base.ProbaQueryStrategy* attribute), 144

`name` (*asreview.models.query.ClusterQuery* attribute), 156

`name` (*asreview.models.query.MaxQuery* attribute), 146

`name` (*asreview.models.query.MaxRandomQuery* attribute), 149

`name` (*asreview.models.query.MaxUncertaintyQuery* attribute), 151

`name` (*asreview.models.query.MixedQuery* property), 147

`name` (*asreview.models.query.RandomQuery* attribute), 154

`name` (*asreview.models.query.UncertaintyQuery* attribute), 153

`name` (*asreview.review.ReviewSimulate* attribute), 106

`NaturePublicationDataGroup` (class in *asreview.datasets*), 99

`NN2LayerClassifier` (class in *asreview.models.classifiers*), 138

`Note`, 5

`notes` (*asreview.ASReviewData* property), 92

## O

`open_state()` (in module *asreview*), 172

## P

`PaperRecord` (class in *asreview.io*), 189

- param (asreview.models.balance.base.BaseBalance property), 159
- param (asreview.models.balance.DoubleBalance property), 162
- param (asreview.models.balance.SimpleBalance property), 160
- param (asreview.models.balance.TripleBalance property), 165
- param (asreview.models.balance.UndersampleBalance property), 166
- param (asreview.models.base.BaseModel property), 107
- param (asreview.models.classifiers.base.BaseTrainClassifier property), 124
- param (asreview.models.classifiers.LogisticClassifier property), 132
- param (asreview.models.classifiers.LSTMBaseClassifier property), 135
- param (asreview.models.classifiers.LSTMPoolClassifier property), 137
- param (asreview.models.classifiers.NaiveBayesClassifier property), 126
- param (asreview.models.classifiers.NN2LayerClassifier property), 140
- param (asreview.models.classifiers.RandomForestClassifier property), 128
- param (asreview.models.classifiers.SVMClassifier property), 130
- param (asreview.models.feature\_extraction.base.BaseFeatureExtractor property), 109
- param (asreview.models.feature\_extraction.Doc2Vec property), 114
- param (asreview.models.feature\_extraction.EmbeddingIdf property), 116
- param (asreview.models.feature\_extraction.EmbeddingLSTM property), 118
- param (asreview.models.feature\_extraction.SBERT property), 121
- param (asreview.models.feature\_extraction.Tfidf property), 111
- param (asreview.models.query.base.BaseQueryStrategy property), 143
- param (asreview.models.query.base.ProbaQueryStrategy property), 144
- param (asreview.models.query.ClusterQuery property), 156
- param (asreview.models.query.MaxQuery property), 146
- param (asreview.models.query.MaxRandomQuery property), 149
- param (asreview.models.query.MaxUncertaintyQuery property), 151
- param (asreview.models.query.MixedQuery property), 147
- param (asreview.models.query.RandomQuery property), 154
- param (asreview.models.query.UncertaintyQuery property), 153
- predict\_proba() (asreview.models.classifiers.base.BaseTrainClassifier method), 125
- predict\_proba() (asreview.models.classifiers.LogisticClassifier method), 133
- predict\_proba() (asreview.models.classifiers.LSTMBaseClassifier method), 136
- predict\_proba() (asreview.models.classifiers.LSTMPoolClassifier method), 138
- predict\_proba() (asreview.models.classifiers.NaiveBayesClassifier method), 127
- predict\_proba() (asreview.models.classifiers.NN2LayerClassifier method), 141
- predict\_proba() (asreview.models.classifiers.RandomForestClassifier method), 129
- predict\_proba() (asreview.models.classifiers.SVMClassifier method), 131
- prior\_data\_idx (asreview.ASReviewData property), 92
- Project file, 5
- Project mode, 5
- Project from id() (in module asreview.project), 183
- Projects dashboard, 5
- ## Q
- query() (asreview.models.query.base.BaseQueryStrategy method), 143
- query() (asreview.models.query.base.ProbaQueryStrategy method), 145
- query() (asreview.models.query.ClusterQuery method), 157
- query() (asreview.models.query.MaxQuery method), 146
- query() (asreview.models.query.MaxRandomQuery method), 150
- query() (asreview.models.query.MaxUncertaintyQuery method), 152
- query() (asreview.models.query.MixedQuery method), 148
- query() (asreview.models.query.RandomQuery method), 155

`query()` (*asreview.models.query.UncertaintyQuery* method), 153

`query_top_ranked()` (*asreview.state.SQLiteState* method), 182

## R

`RandomForestClassifier` (class in *asreview.models.classifiers*), 127

`RandomQuery` (class in *asreview.models.query*), 154

`read_data()` (*asreview.io.CSVReader* class method), 185

`read_data()` (*asreview.io.ExcelReader* class method), 187

`read_data()` (*asreview.io.RISReader* class method), 190

`read_format` (*asreview.io.CSVReader* attribute), 185

`read_format` (*asreview.io.ExcelReader* attribute), 187

`read_format` (*asreview.io.RISReader* attribute), 190

`reader` (*asreview.datasets.BaseDataSet* property), 101

`Record`, 6

`record()` (*asreview.ASReviewData* method), 94

`record_ids` (*asreview.ASReviewData* property), 92

`remove_dataset()` (*asreview.ASReviewProject* method), 171

`remove_error()` (*asreview.ASReviewProject* method), 171

`review()` (*asreview.review.BaseReview* method), 105

`review()` (*asreview.review.ReviewSimulate* method), 106

`Reviewing`, 6

`reviews` (*asreview.ASReviewProject* property), 169

`ReviewSimulate` (class in *asreview.review*), 105

`RISReader` (class in *asreview.io*), 190

`RISWriter` (class in *asreview.io*), 191

## S

`sample()` (*asreview.models.balance.base.BaseBalance* method), 159

`sample()` (*asreview.models.balance.DoubleBalance* method), 163

`sample()` (*asreview.models.balance.SimpleBalance* method), 161

`sample()` (*asreview.models.balance.TripleBalance* method), 165

`sample()` (*asreview.models.balance.UndersampleBalance* method), 167

`SBERT` (class in *asreview.models.feature\_extraction*), 120

`Screeener`, 6

`set_error()` (*asreview.ASReviewProject* method), 171

`settings` (*asreview.review.BaseReview* property), 105

`settings` (*asreview.review.ReviewSimulate* property), 106

`settings` (*asreview.state.SQLiteState* property), 174

`SimpleBalance` (class in *asreview.models.balance*), 160

`SimulateEntryPoint` (class in *asreview.entry\_points*), 197

`SQLiteState` (class in *asreview.state*), 172

`StateInspectEntryPoint` (class in *asreview.entry\_points*), 197

`Status`, 6

`SVMClassifier` (class in *asreview.models.classifiers*), 129

`SynergyDataGroup` (class in *asreview.datasets*), 98

## T

`text` (*asreview.io.PaperRecord* property), 189

`texts` (*asreview.ASReviewData* property), 92

`Tfidf` (class in *asreview.models.feature\_extraction*), 110

`title` (*asreview.ASReviewData* property), 92

`title_length()` (in module *asreview.data.statistics*), 98

`to_dataframe()` (*asreview.ASReviewData* method), 95

`to_dict()` (*asreview.settings.ASReviewSettings* method), 194

`to_dict()` (*asreview.state.SQLiteState* method), 183

`to_file()` (*asreview.ASReviewData* method), 95

`to_file()` (*asreview.datasets.BaseDataSet* method), 101

`train()` (*asreview.review.BaseReview* method), 105

`train()` (*asreview.review.ReviewSimulate* method), 106

`transform()` (*asreview.models.feature\_extraction.base.BaseFeatureExtraction* method), 110

`transform()` (*asreview.models.feature\_extraction.Doc2Vec* method), 115

`transform()` (*asreview.models.feature\_extraction.EmbeddingIdf* method), 117

`transform()` (*asreview.models.feature\_extraction.EmbeddingLSTM* method), 120

`transform()` (*asreview.models.feature\_extraction.SBERT* method), 122

`transform()` (*asreview.models.feature\_extraction.Tfidf* method), 112

`TripleBalance` (class in *asreview.models.balance*), 163

`TSVWriter` (class in *asreview.io*), 192

## U

`UncertaintyQuery` (class in *asreview.models.query*), 152

`UndersampleBalance` (class in *asreview.models.balance*), 166

`update_config()` (*asreview.ASReviewProject* method), 172

`update_decision()` (*asreview.state.SQLiteState* method), 183

`update_review()` (*asreview.ASReviewProject* method), 172

`url` (*asreview.ASReviewData* property), 92

`User`, 6



## V

`version` (*asreview.state.SQLiteState* property), 174

## W

`write_data()` (*asreview.io.CSVWriter* class method),  
186

`write_data()` (*asreview.io.ExcelWriter* class method),  
188

`write_data()` (*asreview.io.RISWriter* class method),  
192

`write_data()` (*asreview.io.TSVWriter* class method),  
193

`write_format` (*asreview.io.CSVReader* attribute), 185

`write_format` (*asreview.io.CSVWriter* attribute), 186

`write_format` (*asreview.io.ExcelReader* attribute), 187

`write_format` (*asreview.io.ExcelWriter* attribute), 188

`write_format` (*asreview.io.RISReader* attribute), 190

`write_format` (*asreview.io.RISWriter* attribute), 191

`write_format` (*asreview.io.TSVWriter* attribute), 193