
pine

JHU/APL

Aug 17, 2020

CONTENTS:

1	API Reference	1
1.1	pine	1
2	Indices and tables	77
	Python Module Index	79
	Index	81

API REFERENCE

This page contains auto-generated API reference documentation¹.

1.1 pine

1.1.1 Subpackages

`pine.backend`

Subpackages

`pine.backend.admin`

This module implements all methods required for user authentication when using PINE's db authentication rather than an external Auth service

Submodules

`pine.backend.admin.bp`

Module Contents

Functions

<code>get_users()</code>	Get the list of all users' details (id, email, and password hash)
<code>get_user(user_id)</code>	Given a user_id, return the user's details (id, email, and password hash)
<code>update_user_password(user_id)</code>	Change the password hash stored in the database for the given user to a newly calculated password hash derived from
<code>update_user(user_id)</code>	Change the details stored in the database for the given user to those provided in the json body of this request.

continues on next page

¹ Created with sphinx-autoapi

Table 1 – continued from previous page

<code>add_user()</code>	Add a new user to PINE, with the details provided in the json body of this request (id, email, and password hash).
<code>delete_user(user_id)</code>	Delete the user matching the given <code>user_id</code>
<code>system_export()</code>	Export the contents of the database as a zip file
<code>system_import()</code>	Import the contents of the data provided in the request body to the database
<code>init_app(app)</code>	

`pine.backend.admin.bp.bp`

`pine.backend.admin.bp.get_users()`

Get the list of all users' details (id, email, and password hash) :return: str

`pine.backend.admin.bp.get_user(user_id)`

Given a `user_id`, return the user's details (id, email, and password hash) :param `user_id`: str :return: str

`pine.backend.admin.bp.update_user_password(user_id)`

Change the password hash stored in the database for the given user to a newly calculated password hash derived from the password provided in the json body of this request. :param `user_id`: :return: Response

`pine.backend.admin.bp.update_user(user_id)`

Change the details stored in the database for the given user to those provided in the json body of this request. :param `user_id`: str :return: Response

`pine.backend.admin.bp.add_user()`

Add a new user to PINE, with the details provided in the json body of this request (id, email, and password hash). This method will calculate and store a password hash based upon the provided password :return: Response

`pine.backend.admin.bp.delete_user(user_id)`

Delete the user matching the given `user_id` :param `user_id`: str :return: Response

`pine.backend.admin.bp.system_export()`

Export the contents of the database as a zip file :return: Response

`pine.backend.admin.bp.system_import()`

Import the contents of the data provided in the request body to the database :return: Response

`pine.backend.admin.bp.init_app(app)`

`pine.backend.annotations`

This module contains the api methods required to perform and display annotations in the front-end and store the annotations in the backend

Submodules

`pine.backend.annotations.bp`

Module Contents

Functions

<code>check_document_by_id(doc_id: str)</code>	Verify that a document with the given doc_id exists and that the logged in user has permissions to access the
<code>check_document(doc: dict)</code>	
<code>get_my_annotations_for_document(doc_id)</code>	Get the list of annotations (key, start_index, end_index) produced by the logged in user for the document matching
<code>get_others_annotations_for_document(doc_id)</code>	Get the list of annotations (key, start_index, end_index) produced by all other users, not including the logged in
<code>get_annotations_for_document(doc_id)</code>	Get the list of annotations (key, start_index, end_index) produced by all users for the document matching the
<code>get_current_annotation(doc_id, user_id)</code>	Get all annotations of the provided document created by the given user.
<code>check_overlapping_annotations(collection, ner_annotations)</code>	
<code>set_document_to_annotated_by_user(doc_id, user_id)</code>	Modify the parameter in the database for the document signifying that the given user has annotated the given
<code>_make_annotations(body)</code>	
<code>_add_or_update_annotation(new_annotation)</code>	
<code>save_annotations(doc_id)</code>	Save new NER annotations and labels to the database as an entry for the logged in user, for the document. If there
<code>save_collection_annotations(collection_id: str)</code>	
<code>init_app(app)</code>	

pine.backend.annotations.bp.logger

pine.backend.annotations.bp.CONFIG_ALLOW_OVERLAPPING_NER_ANNOTATIONS = allow_overlapping_n

pine.backend.annotations.bp.bp

pine.backend.annotations.bp.check_document_by_id (doc_id: str)

Verify that a document with the given doc_id exists and that the logged in user has permissions to access the document :param doc_id: str :return: dict

pine.backend.annotations.bp.check_document (doc: dict)

pine.backend.annotations.bp.get_my_annotations_for_document (doc_id)

Get the list of annotations (key, start_index, end_index) produced by the logged in user for the document matching the provided doc_id. :param doc_id: str :return: Response

pine.backend.annotations.bp.get_others_annotations_for_document (doc_id)

Get the list of annotations (key, start_index, end_index) produced by all other users, not including the logged in user for the document matching the provided doc_id. :param doc_id: str :return: str

pine.backend.annotations.bp.get_annotations_for_document (doc_id)

Get the list of annotations (key, start_index, end_index) produced by all users for the document matching the provided doc_id. :param doc_id: str :return: str

pine.backend.annotations.bp.get_current_annotation (doc_id, user_id)

Get all annotations of the provided document created by the given user. :param doc_id: str :param user_id: str :return: List

pine.backend.annotations.bp.check_overlapping_annotations (collection, ner_annotations)

pine.backend.annotations.bp.set_document_to_annotated_by_user (doc_id, user_id)

Modify the parameter in the database for the document signifying that the given user has annotated the given document :param doc_id: str :param user_id: str :return: whether the update succeeded :rtype: bool

```
pine.backend.annotations.bp._make_annotations(body)
pine.backend.annotations.bp._add_or_update_annotation(new_annotation)
pine.backend.annotations.bp.save_annotations(doc_id)
    Save new NER annotations and labels to the database as an entry for the logged in user, for the document. If
    there are already annotations, use a patch request to update with the new annotations. If there are not, use a post
    request to create a new entry. :param doc_id: str :return: bool
pine.backend.annotations.bp.save_collection_annotations(collection_id: str)
pine.backend.annotations.bp.init_app(app)
```

pine.backend.auth

Submodules

pine.backend.auth.bp

Module Contents

Classes

<code>AuthModule(app, bp)</code>	Initialize self. See help(type(self)) for accurate signature.
----------------------------------	---

Functions

<code>is_flat()</code>
<code>get_logged_in_user()</code>
<code>login_required(view)</code>
<code>admin_required(view)</code>
<code>flask_get_module()</code>
<code>flask_get_flat()</code>
<code>flask_get_can_manage_users()</code>
<code>flask_get_logged_in_user()</code>
<code>flask_get_logged_in_user_details()</code>
<code>flask_get_login_form()</code>
<code>flask_post_logout()</code>
<code>init_app(app)</code>

```
pine.backend.auth.bp.CONFIG_AUTH_MODULE_KEY = AUTH_MODULE
pine.backend.auth.bp.bp
pine.backend.auth.bp.module
pine.backend.auth.bp.is_flat()
pine.backend.auth.bp.get_logged_in_user()
pine.backend.auth.bp.login_required(view)
pine.backend.auth.bp.admin_required(view)
```



```

pine.backend.auth.bp.flask_get_module()
pine.backend.auth.bp.flask_get_flat() → Response
pine.backend.auth.bp.flask_get_can_manage_users() → Response
pine.backend.auth.bp.flask_get_logged_in_user() → Response
pine.backend.auth.bp.flask_get_logged_in_user_details() → Response
pine.backend.auth.bp.flask_get_login_form() → Response
pine.backend.auth.bp.flask_post_logout() → Response
class pine.backend.auth.bp.AuthModule(app, bp)
    Bases: object
    Initialize self. See help(type(self)) for accurate signature.
    __metaclass__
    abstract is_flat(self)
    abstract can_manage_users(self)
    abstract get_login_form(self)
    get_logged_in_user(self)
    get_logged_in_user_details(self)
    logout(self)
pine.backend.auth.bp.init_app(app)

```

pine.backend.auth.eve

Module Contents

Classes

<i>EveUser</i>(data)	Initialize self. See help(type(self)) for accurate signature.
<i>EveModule</i>(app, bp)	

```

class pine.backend.auth.eve.EveUser(data)
    Bases: pine.backend.models.AuthUser
    Initialize self. See help(type(self)) for accurate signature.
    property id(self)
    property username(self)
    property display_name(self)
    property is_admin(self)
    get_details(self)
class pine.backend.auth.eve.EveModule(app, bp)
    Bases: pine.backend.auth.bp.AuthModule
    is_flat(self)

```

```
can_manage_users (self)
get_logged_in_user_details (self)
update_user_details (self)
update_user_password (self)
get_login_form (self)
login (self)
get_all_users (self)
```

`pine.backend.auth.oauth`

Module Contents

Classes

<code>OAuthUser(data)</code>	Initialize self. See <code>help(type(self))</code> for accurate signature.
<code>OAuthModule(app, bp)</code>	

```
class pine.backend.auth.oauth.OAuthUser (data)
    Bases: pine.backend.models.AuthUser
    Initialize self. See help(type(self)) for accurate signature.
    property id (self)
    property username (self)
    property display_name (self)
    property is_admin (self)

class pine.backend.auth.oauth.OAuthModule (app, bp)
    Bases: pine.backend.auth.bp.AuthModule
    abstract register_oauth (self, oauth, app)
    abstract get_login_form_button_text (self)
    is_flat (self)
    can_manage_users (self)
    get_login_form (self)
    login (self)
    authorize (self)
```

`pine.backend.auth.password`

Module Contents

Functions

hash_password(password: str)

check_password(password: str, hashed_password: str)

`pine.backend.auth.password.hash_password` (password: str) → str

`pine.backend.auth.password.check_password` (password: str, hashed_password: str)

`pine.backend.auth.vegas`

Module Contents

Classes

VegasAuthModule(app, bp)

class `pine.backend.auth.vegas.VegasAuthModule` (app, bp)

Bases: *pine.backend.auth.oauth.OAuthModule*

register_oauth (self, oauth, app)

get_login_form_button_text (self)

Package Contents

Functions

login_required(view)

admin_required(view)

get_logged_in_user()

is_flat()

`pine.backend.auth.login_required` (view)

`pine.backend.auth.admin_required` (view)

`pine.backend.auth.module`

`pine.backend.auth.get_logged_in_user` ()

`pine.backend.auth.is_flat` ()

pine.backend.collections

This module contains the api methods required to interact with, organize, create, and display collections in the front-end and store the collections in the backend

Submodules

pine.backend.collections.bp

Module Contents

Functions

<i>is_cached_last_collection(collection_id)</i>	
<i>update_cached_last_collection(collection_id)</i>	
<i>_collection_user_can_projection()</i>	
<i>_collection_user_can(collection, annotate)</i>	
<i>user_can_annotate(collection)</i>	
<i>user_can_view(collection)</i>	
<i>user_can_add_documents_or_images(collection)</i>	
<i>user_can_modify_document_metadata(collection)</i>	
<i>user_can_annotate_by_id(collection_id)</i>	
<i>user_can_annotate_by_ids(collection_ids)</i>	
<i>user_can_view_by_id(collection_id)</i>	
<i>user_can_add_documents_or_images_by_id(collection_id)</i>	
<i>user_can_modify_document_metadata_by_id(collection_id)</i>	
<i>get_user_collections(archived, page)</i>	Return collections for the logged in user using pagination. Returns all collections if parameter “page” is “all”,
<i>get_unarchived_user_collections(page)</i>	Return unarchived user collections for the corresponding page value. Default value returns collections for all
<i>get_archived_user_collections(page)</i>	Return archived user collections for the corresponding page value. Default value returns collections for all
<i>archive_or_unarchive_collection(collection_id, archive)</i>	Set the “archived” boolean flag for the collection matching the provided collection_id.
<i>archive_collection(collection_id)</i>	Archive the collection matching the provided collection id
<i>unarchive_collection(collection_id)</i>	Unarchive the collection matching the provided collection id
<i>get_collection(collection_id)</i>	Return the collection object for the collection matching the provided collection id. This object has the fields:
<i>download_collection(collection_id)</i>	
<i>get_doc_and_overlap_ids(collection_id)</i>	Return lists of ids for overlapping and non-overlapping documents for the collection matching the provided
<i>add_annotator_to_collection(collection_id)</i>	
<i>add_viewer_to_collection(collection_id)</i>	
<i>add_label_to_collection(collection_id)</i>	
<i>get_overlap_ids(collection_id)</i>	Return the list of ids for overlapping documents for the collection matching the provided collection id.
<i>_upload_documents(collection, docs)</i>	

continues on next page

Table 10 – continued from previous page

<code>create_collection()</code>	Create a new collection based upon the entries provided in the POST request's associated form fields.
<code>_check_collection_and_get_image_dir(collection_id, path)</code>	
<code>get_static_collection_images(collection_id)</code>	
<code>get_collection_images(collection_id)</code>	
<code>get_collection_image(collection_id, path)</code>	
<code>get_collection_image_exists(collection_id, path)</code>	
<code>_path_split(path)</code>	
<code>_safe_path(path)</code>	
<code>get_user_can_add_documents_or_images(collection_id)</code>	
<code>_upload_collection_image_file(collection_id, path, image_file)</code>	
<code>post_collection_image(collection_id, path)</code>	
<code>init_app(app)</code>	

`pine.backend.collections.bp.bp`

`pine.backend.collections.bp.LOGGER`

`pine.backend.collections.bp.DOCUMENTS_PER_TRANSACTION = 500`

`pine.backend.collections.bp.LAST_COLLECTION_FOR_IMAGE`

`pine.backend.collections.bp.is_cached_last_collection(collection_id)`

`pine.backend.collections.bp.update_cached_last_collection(collection_id)`

`pine.backend.collections.bp._collection_user_can_projection()`

`pine.backend.collections.bp._collection_user_can(collection, annotate)`

`pine.backend.collections.bp.user_can_annotate(collection)`

`pine.backend.collections.bp.user_can_view(collection)`

`pine.backend.collections.bp.user_can_add_documents_or_images(collection)`

`pine.backend.collections.bp.user_can_modify_document_metadata(collection)`

`pine.backend.collections.bp.user_can_annotate_by_id(collection_id)`

`pine.backend.collections.bp.user_can_annotate_by_ids(collection_ids)`

`pine.backend.collections.bp.user_can_view_by_id(collection_id)`

`pine.backend.collections.bp.user_can_add_documents_or_images_by_id(collection_id)`

`pine.backend.collections.bp.user_can_modify_document_metadata_by_id(collection_id)`

`pine.backend.collections.bp.get_user_collections(archived, page)`

Return collections for the logged in user using pagination. Returns all collections if parameter “page” is “all”, or the collections associated with the given page. Can return archived or un-archived collections based upon the “archived” flag. :param archived: Bool :param page: str :return: Response

`pine.backend.collections.bp.get_unarchived_user_collections(page)`

Return unarchived user collections for the corresponding page value. Default value returns collections for all pages. :param page: str :return: Response

`pine.backend.collections.bp.get_archived_user_collections (page)`
Return archived user collections for the corresponding page value. Default value returns collections for all pages. :param page: str :return: Response

`pine.backend.collections.bp.archive_or_unarchive_collection (collection_id, archive)`
Set the “archived” boolean flag for the collection matching the provided collection_id. :param collection_id: str :param archive: Bool :return: Response

`pine.backend.collections.bp.archive_collection (collection_id)`
Archive the collection matching the provided collection id :param collection_id: str :return: Response

`pine.backend.collections.bp.unarchive_collection (collection_id)`
Unarchive the collection matching the provided collection id :param collection_id: str :return: Response

`pine.backend.collections.bp.get_collection (collection_id)`
Return the collection object for the collection matching the provided collection id. This object has the fields: ‘creator_id’, ‘annotators’, ‘viewers’, ‘labels’, ‘metadata’, ‘archived’, and ‘configuration’. :param collection_id: str :return: Response

`pine.backend.collections.bp.download_collection (collection_id)`

`pine.backend.collections.bp.get_doc_and_overlap_ids (collection_id)`
Return lists of ids for overlapping and non-overlapping documents for the collection matching the provided collection id. :param collection_id: str :return: tuple

`pine.backend.collections.bp.add_annotator_to_collection (collection_id)`

`pine.backend.collections.bp.add_viewer_to_collection (collection_id)`

`pine.backend.collections.bp.add_label_to_collection (collection_id)`

`pine.backend.collections.bp.get_overlap_ids (collection_id)`
Return the list of ids for overlapping documents for the collection matching the provided collection id. :param collection_id: str :return: tuple

`pine.backend.collections.bp._upload_documents (collection, docs)`

`pine.backend.collections.bp.create_collection ()`
Create a new collection based upon the entries provided in the POST request’s associated form fields. These fields include: collection - collection name overlap - ratio of overlapping documents. (0-1) with 0 being no overlap and 1 being every document has overlap, ex:

 .90 - 90% of documents overlap

train_every - automatically train a new classifier after this many documents have been annotated pipelineId - the id value of the classifier pipeline associated with this collection (spacy, opennlp, corenlp) classifierParameters - optional parameters that adjust the configuration of the chosen classifier pipeline. archived - whether or not this collection should be archived. A collection can be created with documents listed in a csv file. Each new line in the csv represents a new document. The data of this csv can be passed to this method through the POST request’s FILES field “file”. used when creating a collection based on an uploaded csv file:

 csvTextCol - column of csv containing the text of the documents (default: 0) csvHasHeader - boolean for whether or not the csv file has a header row (default: False)

A collection can also be created with a number of images through FILES fields “imageFileN” where N is an (ignored) index :return: information about the created collection

`pine.backend.collections.bp._check_collection_and_get_image_dir (collection_id, path)`

`pine.backend.collections.bp.get_static_collection_images (collection_id)`

`pine.backend.collections.bp.get_collection_images (collection_id)`

```

pine.backend.collections.bp.get_collection_image(collection_id, path)
pine.backend.collections.bp.get_collection_image_exists(collection_id, path)
pine.backend.collections.bp._path_split(path)
pine.backend.collections.bp._safe_path(path)
pine.backend.collections.bp.get_user_can_add_documents_or_images(collection_id)
pine.backend.collections.bp._upload_collection_image_file(collection_id, path, image_file)
pine.backend.collections.bp.post_collection_image(collection_id, path)
pine.backend.collections.bp.init_app(app)

```

Package Contents

Functions

<code>user_can_annotate(collection)</code>
<code>user_can_view(collection)</code>
<code>user_can_add_documents_or_images(collection)</code>
<code>user_can_modify_document_metadata(collection)</code>
<code>user_can_annotate_by_id(collection_id)</code>
<code>user_can_annotate_by_ids(collection_ids)</code>
<code>user_can_view_by_id(collection_id)</code>
<code>user_can_add_documents_or_images_by_id(collection_id)</code>
<code>user_can_modify_document_metadata_by_id(collection_id)</code>

```

pine.backend.collections.user_can_annotate(collection)
pine.backend.collections.user_can_view(collection)
pine.backend.collections.user_can_add_documents_or_images(collection)
pine.backend.collections.user_can_modify_document_metadata(collection)
pine.backend.collections.user_can_annotate_by_id(collection_id)
pine.backend.collections.user_can_annotate_by_ids(collection_ids)
pine.backend.collections.user_can_view_by_id(collection_id)
pine.backend.collections.user_can_add_documents_or_images_by_id(collection_id)
pine.backend.collections.user_can_modify_document_metadata_by_id(collection_id)

```

`pine.backend.data`

Submodules

`pine.backend.data.bp`

Module Contents

Functions

`init_app(app)`

`pine.backend.data.bp.init_app(app)`

`pine.backend.data.service`

Module Contents

Classes

<code>PerformanceHistory()</code>	Initialize self. See help(type(self)) for accurate signature.
-----------------------------------	---

Functions

<code>_standardize_path(path, *additional_paths)</code>	
<code>url(path, *additional_paths)</code>	Returns a complete URL for the given eve-relative path(s).
<code>where_params(where)</code>	Returns a “where” parameters object that can be passed to eve.
<code>params(params)</code>	Returns a parameters object that can be passed to eve.
<code>get(path, **kwargs)</code>	Wraps requests.get for the given eve-relative path.
<code>post(path, **kwargs)</code>	Wraps requests.post for the given eve-relative path.
<code>put(path, **kwargs)</code>	Wraps requests.put for the given eve-relative path.
<code>delete(path, **kwargs)</code>	Wraps requests.delete for the given eve-relative path.
<code>patch(path, **kwargs)</code>	Wraps requests.patch for the given eve-relative path.
<code>get_items(path, params={})</code>	
<code>get_item_by_id(path, item_id, params={})</code>	
<code>get_all_versions_of_item_by_id(path, item_id, params={})</code>	
<code>get_all_using_pagination(path, params)</code>	
<code>convert_response(requests_response)</code>	
<code>remove_eve_fields(obj, remove_timestamps=True, remove_versions=True)</code>	
<code>remove_nonupdatable_fields(obj)</code>	

`pine.backend.data.service.logger`

```

class pine.backend.data.service.PerformanceHistory
    Bases: object

    Initialize self. See help(type(self)) for accurate signature.

    pformat (self, **kwargs)

    pprint (self)

    add (self, rest_type, path, response)

pine.backend.data.service.PERFORMANCE_HISTORY
pine.backend.data.service._standardize_path (path, *additional_paths)
pine.backend.data.service.url (path, *additional_paths)
    Returns a complete URL for the given eve-relative path(s).

    Parameters
        • path – str: eve-relative path (e.g. “collections” or [“collections”, id])
        • additional_paths – str[]: any additional paths to append

    Returns str url

pine.backend.data.service.where_params (where)
    Returns a “where” parameters object that can be passed to eve.

    Eve requires that dict parameters be serialized as JSON.

    Parameters where – dict: dictionary of “where” params to pass to eve

    Returns dict “where” params

pine.backend.data.service.params (params)
    Returns a parameters object that can be passed to eve.

    Eve requires that dict parameters be serialized as JSON.

    Parameters where – dict: dictionary of “where” params to pass to eve

    Returns dict “where” params

pine.backend.data.service.get (path, **kwargs)
    Wraps requests.get for the given eve-relative path.

    Parameters
        • path – str: eve-relative path (e.g. “collections” or [“collections”, id])
        • **kwargs – dict: any additional arguments to pass to requests.get

    Returns requests.Response

pine.backend.data.service.post (path, **kwargs)
    Wraps requests.post for the given eve-relative path.

    Parameters
        • path – str: eve-relative path (e.g. “collections” or [“collections”, id])
        • **kwargs – dict: any additional arguments to pass to requests.post

    Returns requests.Response

pine.backend.data.service.put (path, **kwargs)
    Wraps requests.put for the given eve-relative path.

```

Parameters

- **path** – str: eve-relative path (e.g. “collections” or [“collections”, id])
- ****kwargs** – dict: any additional arguments to pass to requests.put

Returns requests.Response`pine.backend.data.service.delete(path, **kwargs)`

Wraps requests.delete for the given eve-relative path.

Parameters

- **path** – str: eve-relative path (e.g. “collections” or [“collections”, id])
- ****kwargs** – dict: any additional arguments to pass to requests.delete

Returns requests.Response`pine.backend.data.service.patch(path, **kwargs)`

Wraps requests.patch for the given eve-relative path.

Parameters

- **path** – str: eve-relative path (e.g. “collections” or [“collections”, id])
- ****kwargs** – dict: any additional arguments to pass to requests.patch

Returns requests.Response`pine.backend.data.service.get_items(path, params={})``pine.backend.data.service.get_item_by_id(path, item_id, params={})``pine.backend.data.service.get_all_versions_of_item_by_id(path, item_id, params={})``pine.backend.data.service.get_all_using_pagination(path, params)``pine.backend.data.service.convert_response(requests_response)``pine.backend.data.service.remove_eve_fields(obj, remove_timestamps=True, remove_versions=True)``pine.backend.data.service.remove_nonupdatable_fields(obj)`**pine.backend.data.users****Module Contents****Functions**

`get_all_users()`

`get_user(user_id)`

`get_user_by_email(email)`

`get_user_details(user_id)`

`update_user(user_id: str, details: models.UserDetails)`

`print_users_command()`

`add_admin_command(username, password)`

`set_user_password_by_id(user_id, password)`

`set_user_password(username, password)`

continues on next page

Table 15 – continued from previous page

`reset_user_passwords()`

```

pine.backend.data.users.get_all_users()
pine.backend.data.users.get_user(user_id)
pine.backend.data.users.get_user_by_email(email)
pine.backend.data.users.get_user_details(user_id)
pine.backend.data.users.update_user(user_id: str, details: models.UserDetails)
pine.backend.data.users.print_users_command()
pine.backend.data.users.add_admin_command(username, password)
pine.backend.data.users.set_user_password_by_id(user_id, password)
pine.backend.data.users.set_user_password(username, password)
pine.backend.data.users.reset_user_passwords()

```

pine.backend.documents

Submodules

pine.backend.documents.bp

Module Contents

Functions

```

_document_user_can_projection()
get_collection_ids_for(document_ids)
user_can_annotate(document)
user_can_view(document)
user_can_modify_metadata(document)
user_can_annotate_by_id(document_id)
user_can_annotate_by_ids(document_ids)
user_can_view_by_id(document_id)
user_can_modify_metadata_by_id(document_id)
get_document(doc_id)
get_documents_in_collection(col_id, page)
_check_documents(documents)
add_document()
can_annotate_document(doc_id)
can_modify_metadata(doc_id)
update_metadata(doc_id)
init_app(app)

```

```
pine.backend.documents.bp.bp
```

```
pine.backend.documents.bp._document_user_can_projection()
```

```
pine.backend.documents.bp.get_collection_ids_for(document_ids) → set
pine.backend.documents.bp.user_can_annotate(document)
pine.backend.documents.bp.user_can_view(document)
pine.backend.documents.bp.user_can_modify_metadata(document)
pine.backend.documents.bp.user_can_annotate_by_id(document_id)
pine.backend.documents.bp.user_can_annotate_by_ids(document_ids)
pine.backend.documents.bp.user_can_view_by_id(document_id)
pine.backend.documents.bp.user_can_modify_metadata_by_id(document_id)
pine.backend.documents.bp.get_document(doc_id)
pine.backend.documents.bp.get_documents_in_collection(col_id, page)
pine.backend.documents.bp._check_documents(documents) → dict
pine.backend.documents.bp.add_document()
pine.backend.documents.bp.can_annotate_document(doc_id)
pine.backend.documents.bp.can_modify_metadata(doc_id)
pine.backend.documents.bp.update_metadata(doc_id)
pine.backend.documents.bp.init_app(app)
```

pine.backend.job_manager

Submodules

pine.backend.job_manager.service

Module Contents

Classes

ServiceManager(default_handler=None)

type default_handler callable

pine.backend.job_manager.service.config

pine.backend.job_manager.service.logger

class pine.backend.job_manager.service.**ServiceManager** (default_handler=None)

Bases: *object*

r_pool

r_conn

redis_key_prefix

redis_reg_key_prefix

```

redis_channels_key
redis_channel_ttl_key_prefix
redis_work_queue_key_prefix
redis_work_mutex_key_prefix
redis_handler_mutex_key_prefix
redis_reg_key_ttl
redis_channels_key_ttl
redis_work_queue_key_ttl
redis_work_mutex_key_ttl
redis_handler_mutex_key_ttl
handler_timeout
registration_worker_name = registration_worker
processing_worker_name = processing_worker
channel_worker_name = channel_worker
shutdown_channel = shutdown
registration_channel = registration
reserved_channels

classmethod get_registered_channels (cls, include_ttl=False)
    Get list of registered channels, with registration time if requested. :type include_ttl: bool :rtype: list[str] |
    dict[str, datetime]

classmethod get_registered_service_details (cls, service_name=None)
    Get registration details of a service. :type service_name: str :rtype: None | dict

classmethod get_registered_services (cls, include_details=True)
    Get list of registered services and registration body if requested. :type include_details: bool :rtype: list[str]
    | list[dict]

classmethod send_service_request (cls, service_name, data, job_id=None, encoder=None)
    Queue's a job for the requested service. :type service_name: str :type data: dict :type job_id: str :type
    encoder: json.JSONEncoder :rtype: None | dict

start_listeners (self)
    Starts all the workers.

stop_listeners (self)
    Stops all the workers.

_start_registration_listener (self)
    Starts the registration worker. :rtype: bool

_start_processing_listeners (self)
    Starts the processing workers. :rtype: bool

_start_channel_watchdog (self)
    Starts the channel watchdog workers in an asyncio-only thread. It monitors the channel TTL's. :rtype:
    bool

_stop_channel_watchdog (self)
    Stops the channel watchdog workers in an asyncio-only thread. :rtype: bool

```

`_registration_listener` (*self*)

Registration Listener Implementation.

`async _channel_watchdog` (*self*)

Channel Watchdog Implementation. In asyncio, it monitors the channel-ttl keys and the channels SET to expire registered services as needed. The other functionality is to register new channels as they're added to the pubsub in the processing listener.

`static _thread_killer` (*thread_id*)

`_processing_listener_handler_wrapper` (*self, job_id, job_type, job_queue, job_data*)

`_processing_listener` (*self*)

Processing Listener Implementation. Runs a handler when a processing message gets send over an already registered channel.

`pine.backend.job_manager.service.sm`

`pine.backend.pineiaa`

Subpackages

`pine.backend.pineiaa.bratiaa`

Submodules

`pine.backend.pineiaa.bratiaa.agree`

Module Contents

Classes

<code>Document</code> (txt, doc_id)	Initialize self. See help(type(self)) for accurate signature.
<code>FlAgreement</code> (annotators, documents, labels, eval_func=exact_match_instance_evaluation, to-ken_func=None)	Initialize self. See help(type(self)) for accurate signature.

Functions

<code>input_generator</code> (json_list)
<code>compute_fl</code> (tp, fp, fn)
<code>compute_fl_agreement</code> (annotators, documents, labels, token_func=None, eval_func=None)
<code>iaa_report</code> (fl_agreement, precision=3)

`pine.backend.pineiaa.bratiaa.agree.Annotation`

`pine.backend.pineiaa.bratiaa.agree.AnnFile`

`pine.backend.pineiaa.bratiaa.agree.LOGGER`

`class` `pine.backend.pineiaa.bratiaa.agree.Document` (*txt, doc_id*)

Initialize self. See help(type(self)) for accurate signature.

```
__slots__ = ['ann_files', 'txt', 'doc_id']
```

```
pine.backend.pineiaa.bratiaa.agree.input_generator (json_list)
```

```
pine.backend.pineiaa.bratiaa.agree.compute_f1 (tp, fp, fn)
```

```
class pine.backend.pineiaa.bratiaa.agree.F1Agreement (annotators, documents, labels,
                                                    eval_func=exact_match_instance_evaluation,
                                                    token_func=None)
```

Initialize self. See help(type(self)) for accurate signature.

```
property annotators (self)
```

```
property documents (self)
```

```
property labels (self)
```

```
_compute_tp_fp_fn (self, documents)
```

```
_increment_counts (self, annotations, pair, doc, kind)
```

```
mean_sd_per_label (self)
```

Mean and standard deviation of all annotator combinations' F1 scores by label.

```
mean_sd_per_document (self)
```

Mean and standard deviation of all annotator combinations' F1 scores per document.

```
mean_sd_total (self)
```

Mean and standard deviation of all annotator combinations' F1 scores.

```
mean_sd_per_label_one_vs_rest (self, annotator)
```

Mean and standard deviation of all annotator combinations' F1 scores involving given annotator per label.

```
mean_sd_total_one_vs_rest (self, annotator)
```

Mean and standard deviation of all annotator combinations' F1 scores involving given annotator.

```
_pairs_involving (self, annotator)
```

```
static _mean_sd (f1_pairs)
```

Mean and standard deviation along first axis.

```
static print_table (row_label_header, row_labels, avg, stddev, precision=3)
```

```
compute_total_f1_matrix (self)
```

Returns (n x n) matrix, where n is the number of annotators, containing pair-wise total F1 scores between all annotators.

By definition, the matrix is symmetric and $F1 = 1$ on the main diagonal.

```
draw_heatmap (self, out_path)
```

Draws heatmap based on square matrix of F1 scores.

```
pine.backend.pineiaa.bratiaa.agree.compute_f1_agreement (annotators, documents,
                                                         labels, token_func=None,
                                                         eval_func=None)
```

```
pine.backend.pineiaa.bratiaa.agree.iaa_report (f1_agreement, precision=3)
```

`pine.backend.pineiaa.bratiaa.agree_cli`

Module Contents

Functions

`parse_args()`

`main()`

`pine.backend.pineiaa.bratiaa.agree_cli.parse_args()`

`pine.backend.pineiaa.bratiaa.agree_cli.main()`

`pine.backend.pineiaa.bratiaa.evaluation`

Functions for computing the difference between two sets of annotations.

Module Contents

Functions

`exact_match_instance_evaluation(ann_list_1,
ann_list2, tokens=None)`

`exact_match_token_evaluation(ann_list_1,` Annotations are split into token-sized bits before true posi-
`ann_list_2, tokens=None)` tives, false positives and false negatives are computed.

`counter2list(c)`

`_read_token_annotations(ann_list, tokens)` Yields a new annotation for each token overlapping with an
annotation. If annotations are overlapping each other,

`pine.backend.pineiaa.bratiaa.evaluation.Annotation`

`pine.backend.pineiaa.bratiaa.evaluation.exact_match_instance_evaluation` (*ann_list_1*,
ann_list2,
to-
kens=None)

`pine.backend.pineiaa.bratiaa.evaluation.exact_match_token_evaluation` (*ann_list_1*,
ann_list_2,
to-
kens=None)

Annotations are split into token-sized bits before true positives, false positives and false negatives are computed.

Sub-token annotations are expanded to full tokens. Long annotations will influence the results more than short annotations. Boundary errors for adjacent annotations with the same label are ignored!

`pine.backend.pineiaa.bratiaa.evaluation.counter2list` (*c*)

`pine.backend.pineiaa.bratiaa.evaluation._read_token_annotations` (*ann_list*,
tokens)

Yields a new annotation for each token overlapping with an annotation. If annotations are overlapping each other, there will be multiple annotations for a single token.

`pine.backend.pineiaa.bratiaa.iaa_service`

Module Contents

Functions

<code>get_items(resource)</code>	
<code>get_all_items(query)</code>	
<code>get_doc_annotations(collection_id,</code>	ex-
<code>clude=None)</code>	
<code>fix_num_for_json(number)</code>	
<code>getIAAReportForCollection(collection_id)</code>	

`pine.backend.pineiaa.bratiaa.iaa_service.EVE_HEADERS`

`pine.backend.pineiaa.bratiaa.iaa_service.get_items(resource)`

`pine.backend.pineiaa.bratiaa.iaa_service.get_all_items(query)`

`pine.backend.pineiaa.bratiaa.iaa_service.get_doc_annotations(collection_id,`
`exclude=None)`

`pine.backend.pineiaa.bratiaa.iaa_service.fix_num_for_json(number)`

`pine.backend.pineiaa.bratiaa.iaa_service.getIAAReportForCollection(collection_id)`

`pine.backend.pineiaa.bratiaa.utils`

Module Contents

Classes

<code>TokenOverlap(text, tokens)</code>	Data structure for quick lookup of tokens overlapping with given span.
---	--

Functions

<code>tokenize(text)</code>	
<code>read(path,</code>	<code>encoding=ENCODING,</code>
<code>mode='r')</code>	<code>newline='\r\n',</code>

`pine.backend.pineiaa.bratiaa.utils.ENCODING = utf-8`

`pine.backend.pineiaa.bratiaa.utils.TOKEN`

`pine.backend.pineiaa.bratiaa.utils.tokenize(text)`

`pine.backend.pineiaa.bratiaa.utils.read(path,` `encoding=ENCODING,` `newline='\r\n',`
`mode='r')`

class `pine.backend.pineiaa.bratiaa.utils.TokenOverlap(text, tokens)`

Data structure for quick lookup of tokens overlapping with given span. Assumes that the provided list of tokens

is sorted by indices!

Initialize self. See help(type(self)) for accurate signature.

static compute_mapping (*text_length, tokens*)

overlapping_tokens (*self, start, end*)

Package Contents

Classes

<i>F1Agreement</i> (annotators, documents, labels, eval_func=exact_match_instance_evaluation, ken_func=None)	Initialize self. See help(type(self)) for accurate signature.
<i>Document</i> (txt, doc_id)	Initialize self. See help(type(self)) for accurate signature.

Functions

<i>compute_f1_agreement</i> (annotators, documents, labels, token_func=None, eval_func=None)	
<i>iaa_report</i> (f1_agreement, precision=3)	
<i>input_generator</i> (json_list)	
<i>exact_match_instance_evaluation</i> (ann_list_1, ann_list2, tokens=None)	
<i>exact_match_token_evaluation</i> (ann_list_1, ann_list_2, tokens=None)	Annotations are split into token-sized bits before true positives, false positives and false negatives are computed.
<i>tokenize</i> (text)	

pine.backend.pineiaa.bratiaa.**compute_f1_agreement** (*annotators, documents, labels, ken_func=None, eval_func=None*)

pine.backend.pineiaa.bratiaa.**iaa_report** (*f1_agreement, precision=3*)

pine.backend.pineiaa.bratiaa.**AnnFile**

class pine.backend.pineiaa.bratiaa.**F1Agreement** (*annotators, documents, labels, eval_func=exact_match_instance_evaluation, token_func=None*)

Initialize self. See help(type(self)) for accurate signature.

property **annotators** (*self*)

property **documents** (*self*)

property **labels** (*self*)

_compute_tp_fp_fn (*self, documents*)

_increment_counts (*self, annotations, pair, doc, kind*)

mean_sd_per_label (*self*)

Mean and standard deviation of all annotator combinations' F1 scores by label.

mean_sd_per_document (*self*)

Mean and standard deviation of all annotator combinations' F1 scores per document.

```

mean_sd_total (self)
    Mean and standard deviation of all annotator combinations' F1 scores.

mean_sd_per_label_one_vs_rest (self, annotator)
    Mean and standard deviation of all annotator combinations' F1 scores involving given annotator per label.

mean_sd_total_one_vs_rest (self, annotator)
    Mean and standard deviation of all annotator combinations' F1 scores involving given annotator.

_pairs_involving (self, annotator)

static _mean_sd (fl_pairs)
    Mean and standard deviation along first axis.

static print_table (row_label_header, row_labels, avg, stddev, precision=3)

compute_total_f1_matrix (self)
    Returns (n x n) matrix, where n is the number of annotators, containing pair-wise total F1 scores between
    all annotators.

    By definition, the matrix is symmetric and F1 = 1 on the main diagonal.

draw_heatmap (self, out_path)
    Draws heatmap based on square matrix of F1 scores.

class pine.backend.pineiaa.bratiaa.Document (txt, doc_id)
    Initialize self. See help(type(self)) for accurate signature.

    __slots__ = ['ann_files', 'txt', 'doc_id']

pine.backend.pineiaa.bratiaa.input_generator (json_list)

pine.backend.pineiaa.bratiaa.exact_match_instance_evaluation (ann_list_1,
                                                             ann_list2,          to-
                                                             tokens=None)

pine.backend.pineiaa.bratiaa.exact_match_token_evaluation (ann_list_1, ann_list_2,
                                                            tokens=None)
    Annotations are split into token-sized bits before true positives, false positives and false negatives are computed.

    Sub-token annotations are expanded to full tokens. Long annotations will influence the results more than short
    annotations. Boundary errors for adjacent annotations with the same label are ignored!

pine.backend.pineiaa.bratiaa.Annotation

pine.backend.pineiaa.bratiaa.tokenize (text)

```

Submodules

pine.backend.pineiaa.bp

Module Contents

Functions

```

get_current_report(collection_id)
get_iaa_report_by_collection_id(collection_id)
create_iaa_report_by_collection_id(collection_id)
init_app(app)

```

```
pine.backend.pineiaa.bp.logger
pine.backend.pineiaa.bp.bp
pine.backend.pineiaa.bp.get_current_report(collection_id)
pine.backend.pineiaa.bp.get_iaa_report_by_collection_id(collection_id)
pine.backend.pineiaa.bp.create_iaa_report_by_collection_id(collection_id)
pine.backend.pineiaa.bp.init_app(app)
```

pine.backend.pipelines

Submodules

pine.backend.pipelines.bp

Module Contents

Functions

<i>get_pipelines()</i>
<i>get_pipeline_by_id(pipeline_id)</i>
<i>_get_collection_classifier(collection_id)</i>
<i>get_collection_classifier(collection_id)</i>
<i>_get_classifier_metrics(classifier_id)</i>
<i>get_metrics()</i>
<i>get_classifier_metrics(classifier_id)</i>
<i>_get_classifier(classifier_id)</i>
<i>_get_next_instance(classifier_id)</i>
<i>get_next_by_classifier(classifier_id)</i>
<i>advance_to_next_document_by_classifier(classifier_id,</i>
<i>document_id)</i>
<i>predict()</i>
<i>test_redis()</i>
<i>init_app(app)</i>

```
pine.backend.pipelines.bp.logger
pine.backend.pipelines.bp.service_manager
pine.backend.pipelines.bp.bp
pine.backend.pipelines.bp.classifier_dict
pine.backend.pipelines.bp.classifier_pipelines
pine.backend.pipelines.bp.get_pipelines()
pine.backend.pipelines.bp.get_pipeline_by_id(pipeline_id)
pine.backend.pipelines.bp._get_collection_classifier(collection_id)
pine.backend.pipelines.bp.get_collection_classifier(collection_id)
pine.backend.pipelines.bp._get_classifier_metrics(classifier_id)
```

```

pine.backend.pipelines.bp.get_metrics()
pine.backend.pipelines.bp.get_classifier_metrics(classifier_id)
pine.backend.pipelines.bp._get_classifier(classifier_id)
pine.backend.pipelines.bp._get_next_instance(classifier_id)
pine.backend.pipelines.bp.get_next_by_classifier(classifier_id)
pine.backend.pipelines.bp.advance_to_next_document_by_classifier(classifier_id,
                                                                    docu-
                                                                    ment_id)

pine.backend.pipelines.bp.predict()
pine.backend.pipelines.bp.test_redis()
pine.backend.pipelines.bp.init_app(app)

```

pine.backend.shared

Submodules

pine.backend.shared.config

Module Contents

Classes

<code>BaseConfig</code> (root_dir=None)	Initialize self. See help(type(self)) for accurate signature.
<code>TestConfig</code> (root_dir=None)	Initialize self. See help(type(self)) for accurate signature.
<code>ConfigBuilder</code> ()	Initialize self. See help(type(self)) for accurate signature.

pine.backend.shared.config.LOGGER

class pine.backend.shared.config.**BaseConfig** (root_dir=None)

Bases: `object`

Initialize self. See help(type(self)) for accurate signature.

ROOT_DIR

BASE_DIR

BASE_CFG_FILE = `config.yaml`

BASE_ENV_PREFIX = `AL_`

DEBUG = `True`

TESTING = `False`

LOGGER_NAME

LOGGER_DIR = `logs`

LOGGER_FILE = `debug.log`

LOGGER_LEVEL

```
EVE_HOST = localhost
EVE_PORT = 5001
REDIS_HOST = localhost
REDIS_PORT = 6379
REDIS_USR
REDIS_PWD
REDIS_DBNUM = 0
REDIS_PREFIX = AL:
REDIS_EXPIRE = 3600
SCHEDULER_REGISTRATION_TIMEOUT
SCHEDULER_HANDLER_TIMEOUT
SCHEDULER_QUEUE_TIMEOUT
SERVICE_REGISTRATION_CHANNEL = registration
SERVICE_REGISTRATION_FREQUENCY = 60
SERVICE_LISTENING_FREQUENCY = 1
SERVICE_HANDLER_TIMEOUT = 3600
SERVICE_LIST
DATASETS_LOCAL_DIR = D:\Data\DEEPCATT2_DB_DATA\datasets
MODELS_LOCAL_DIR = D:\\Data\\DEEPCATT2_DB_DATA\\Models
classmethod _get_config_var_paths(cls, root_dict=None)
classmethod _process_paths(cls, alt_path=None)
classmethod _process_file_cfg(cls)
classmethod _process_env_vars(cls)
classmethod as_dict(cls)
    Return type dict
classmethod as_attr_dict(cls)
    Return type munch.Munch | dict
static _try_cast(value, _type, _default=None)
static _str2bool(_str, _default=None)
class pine.backend.shared.config.TestConfig(root_dir=None)
    Bases: pine.backend.shared.config.BaseConfig
    Initialize self. See help(type(self)) for accurate signature.
class pine.backend.shared.config.ConfigBuilder
    Bases: object
    Initialize self. See help(type(self)) for accurate signature.
    __env_cfg_variable = BUILDER_CFG_PROFILE
    __current_config_instance
```

```

__current_config_instance_name
__current_config_instance_print = False
__arg_parser
static __get_configs()
    :rtype list[Callable[... BaseConfig]]
static get_config_names()
    Return type list[str]
classmethod get_arg_parser(cls)
    Return type ArgumentParser
classmethod init_config(cls, config_name=None, config_base=None, enable_terminal=True,
    as_attr_dict=True)
classmethod get_config(cls, config_name=None, config_base=None, enable_terminal=True,
    as_attr_dict=True)
    Return type BaseConfig
classmethod set_config(cls, config_name=None, config_base=None, enable_terminal=True,
    as_attr_dict=True)
classmethod __parse_terminal_config(cls)
    Return type argparse.Namespace

```

`pine.backend.shared.transform`

Module Contents

Functions

<code>transform_module_by_config(module_ref, config_ref, config_prefix=None)</code>	Transforms a given module's properties based on ConfigBuilder Values.
---	---

`pine.backend.shared.transform.transform_module_by_config(module_ref, config_ref, config_prefix=None)`

Transforms a given module's properties based on ConfigBuilder Values. The prefix can be used to avoid blindly changing values and target a subset of matching values in config_ref. :type module_ref: ModuleType :type config_ref: dict :type config_prefix: str

Submodules

pine.backend.app

Module Contents

Functions

<i>handle_error</i> (e)
<i>handle_uncaught_exception</i> (e)
<i>create_app</i> (test_config=None)

pine.backend.app.VERSION
pine.backend.app.LOGGER
pine.backend.app.handle_error(e)
pine.backend.app.handle_uncaught_exception(e)
pine.backend.app.create_app(test_config=None)

pine.backend.config

Module Contents

pine.backend.config.SECRET_KEY = Cq13XII=%
pine.backend.config.DEBUG = True
pine.backend.config.EVE_SERVER
pine.backend.config.REDIS_SERVER
pine.backend.config.REDIS_PORT
pine.backend.config.AUTH_MODULE
pine.backend.config.AUTH_MODULE = vegas
pine.backend.config.VEGAS_CLIENT_SECRET
pine.backend.config.DOCUMENT_IMAGE_DIR

pine.backend.cors

Module Contents

Functions

<i>not_found</i> (e)
<i>init_app</i> (app)

```
pine.backend.cors.not_found(e)
pine.backend.cors.init_app(app)
```

pine.backend.log

Module Contents

Classes

<i>Action</i> ()	Generic enumeration.
------------------	----------------------

Functions

<i>setup_logging</i> ()
<i>get_flask_request_info</i> ()
<i>get_flask_logged_in_user</i> ()
<i>access_flask_login</i> ()
<i>access_flask_logout</i> (user: dict)
<i>access_flask_add_collection</i> (collection: dict)
<i>access_flask_view_document</i> (document: dict)
<i>access_flask_add_document</i> (document: dict)
<i>access_flask_add_documents</i> (documents: typing.List[dict])
<i>access_flask_annotate_document</i> (annotation)
<i>access_flask_annotate_documents</i> (annotations: typing.List[dict])
<i>access</i> (action, user, request_info, message, **extra_info)

```
pine.backend.log.CONFIG_FILE_ENV = PINE_LOGGING_CONFIG_FILE
```

```
pine.backend.log.ACCESS_LOGGER_NAME = pine.access
```

```
pine.backend.log.ACCESS_LOGGER
```

```
class pine.backend.log.Action
```

Bases: `enum.Enum`

Generic enumeration.

Derive from this class to define new enumerations.

Create and return a new object. See `help(type)` for accurate signature.

LOGIN

LOGOUT

CREATE_COLLECTION

VIEW_DOCUMENT

ADD_DOCUMENT

ADD_DOCUMENTS**ANNOTATE_DOCUMENT****ANNOTATE_DOCUMENTS**

```
pine.backend.log.setup_logging()
pine.backend.log.get_flask_request_info()
pine.backend.log.get_flask_logged_in_user()
pine.backend.log.access_flask_login()
pine.backend.log.access_flask_logout(user: dict)
pine.backend.log.access_flask_add_collection(collection: dict)
pine.backend.log.access_flask_view_document(document: dict)
pine.backend.log.access_flask_add_document(document: dict)
pine.backend.log.access_flask_add_documents(documents: typing.List[dict])
pine.backend.log.access_flask_annotate_document(annotation)
pine.backend.log.access_flask_annotate_documents(annotations: typing.List[dict])
pine.backend.log.access(action, user, request_info, message, **extra_info)
```

pine.backend.models

Module Contents

Classes

<code>LoginFormFieldType()</code>	Generic enumeration.
<code>LoginFormField(name: str, display: str, field_type: LoginFormFieldType)</code>	Initialize self. See help(type(self)) for accurate signature.
<code>LoginForm(fields: list, button_text: str)</code>	Initialize self. See help(type(self)) for accurate signature.
<code>AuthUser()</code>	Initialize self. See help(type(self)) for accurate signature.
<code>UserDetails(first_name, last_name, description)</code>	Initialize self. See help(type(self)) for accurate signature.

class pine.backend.models.LoginFormFieldTypeBases: `enum.Enum`

Generic enumeration.

Derive from this class to define new enumerations.

Create and return a new object. See help(type) for accurate signature.

TEXT = text**PASSWORD = password****class** pine.backend.models.LoginFormField(name: *str*, display: *str*, field_type: LoginFormFieldType)Bases: `object`

Initialize self. See help(type(self)) for accurate signature.

```

    to_dict (self)
class pine.backend.models.LoginForm (fields: list, button_text: str)
    Bases: object
    Initialize self. See help(type(self)) for accurate signature.
    to_dict (self)
class pine.backend.models.AuthUser
    Bases: object
    Initialize self. See help(type(self)) for accurate signature.
    property id (self)
    property username (self)
    property display_name (self)
    property is_admin (self)
    to_dict (self)
class pine.backend.models.UserDetails (first_name, last_name, description)
    Bases: object
    Initialize self. See help(type(self)) for accurate signature.
    to_dict (self)

```

Package Contents

Functions

```
create_app(test_config=None)
```

```

pine.backend.create_app (test_config=None)
pine.backend.VERSION
pine.backend.__version__

```

pine.client

PINE client module.

Submodules

pine.client.client

PINE client classes module.

Module Contents

Classes

<code>BaseClient</code> (base_uri: str, name: str = None)	Base class for a client using a REST interface.
<code>EveClient</code> (eve_base_uri: str, mongo_base_uri: str = None, mongo_dbname: str = DEFAULT_DBNAME)	A client to access Eve and, optionally, its underlying MongoDB instance.
<code>PineClient</code> (backend_base_uri: str)	A client to access PINE (more specifically: the backend).
<code>LocalPineClient</code> (backend_base_uri: str, eve_base_uri: str, mongo_base_uri: str = None, mongo_dbname: str = EveClient.DEFAULT_DBNAME)	A client for a local PINE instance, including an :py:class<EveClient>.

Functions

<code>_standardize_path</code> (path: str, *additional_paths: typing.List[str])	Standardize path(s) into a list of path components.
---	---

`pine.client.client._standardize_path` (path: str, *additional_paths: typing.List[str]) → typing.List[str]

Standardize path(s) into a list of path components.

Parameters

- **path** (str) – relative path, e.g. "users"
- ***additional_paths** (list(str), optional) – any additional path components in a list

Returns the standardized path components in a list

Return type list(str)

class `pine.client.client.BaseClient` (base_uri: str, name: str = None)

Bases: `object`

Base class for a client using a REST interface.

Constructor.

Parameters

- **base_uri** (str) – the base URI for the server, e.g. "http://localhost:5000"
- **name** (str, optional) – optional human-readable name for the server, defaults to None

`__metaclass__`

base_uri :str

The server's base URI.

Type str

session :requests.Session

The currently open session, or None.

Type requests.Session

abstract is_valid(*self*)

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type `bool`

uri(*self*, *path*: `str`, **additional_paths*: `typing.List[str]`)

Makes a complete URI from the given path(s).

Parameters

- **path** (`str`) – relative path, e.g. "users"
- ***additional_paths** (`list(str)`, *optional*) – any additional path components

Returns the complete, standardized URI including the base URI, e.g. "http://localhost:5000/users"

Return type `str`

_req(*self*, *method*: `str`, *path*: `str`, **additional_paths*: `typing.List[str]`, ***kwargs*)

Makes a `requests` call, checks for errors, and returns the response.

Parameters

- **method** (`str`) – REST method ("get", "post", etc.)
- **path** (`str`) – relative path, e.g. "users"
- ***additional_paths** (`list(str)`, *optional*) – any additional path components
- ****kwargs** (`dict`) – any additional kwargs to send to `requests`

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the `requests requests.Response` object

Return type `requests.Response`

get(*self*, *path*: `str`, **additional_paths*: `typing.List[str]`, ***kwargs*)

Makes a `requests` GET call, checks for errors, and returns the response.

Parameters

- **path** (`str`) – relative path, e.g. "users"
- ***additional_paths** (`list(str)`, *optional*) – any additional path components
- ****kwargs** (`dict`) – any additional kwargs to send to `requests`

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the `requests Response` object

Return type `requests.Response`

put(*self*, *path*: `str`, **additional_paths*: `typing.List[str]`, ***kwargs*)

Makes a `requests` PUT call, checks for errors, and returns the response.

Parameters

- **path** (`str`) – relative path, e.g. "users"

- ***additional_paths** (*list(str), optional*) – any additional path components
- ****kwargs** (*dict*) – any additional kwargs to send to `requests`

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the `requests` `Response` object

Return type `requests.Response`

patch (*self, path: str, *additional_paths: typing.List[str], **kwargs*)

Makes a `requests` PATCH call, checks for errors, and returns the response.

Parameters

- **path** (*str*) – relative path, e.g. "users"
- ***additional_paths** (*list(str), optional*) – any additional path components
- ****kwargs** (*dict*) – any additional kwargs to send to `requests`

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the `requests` `Response` object

Return type `requests.Response`

post (*self, path: str, *additional_paths: typing.List[str], **kwargs*)

Makes a `requests` POST call, checks for errors, and returns the response.

Parameters

- **path** (*str*) – relative path, e.g. "users"
- ***additional_paths** (*list(str), optional*) – any additional path components
- ****kwargs** (*dict*) – any additional kwargs to send to `requests`

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the `requests` `Response` object

Return type `requests.Response`

class `pine.client.client.EveClient` (*eve_base_uri: str, mongo_base_uri: str = None, mongo_dbname: str = DEFAULT_DBNAME*)

Bases: `pine.client.client.BaseClient`

A client to access Eve and, optionally, its underlying MongoDB instance.

Constructor.

Parameters

- **eve_base_uri** (*str*) – the base URI for the eve server, e.g. "http://localhost:5001"
- **mongo_base_uri** (*str, optional*) – the base URI for the mongodb server, e.g. "mongodb://localhost:27018", defaults to None
- **mongo_dbname** (*str, optional*) – the DB name that PINE uses, defaults to "pmap_nlp"

DEFAULT_DBNAME :str = `pmap_nlp`

The default DB name used by PINE.

Type str

mongo_base_uri :str

The base URI for the MongoDB server.

Type str

mongo :pymongo.MongoClient

The `pymongo.mongo_client.MongoClient` instance.

Type pymongo.mongo_client.MongoClient

mongo_db :pymongo.database.Database

The `pymongo.database.Database` instance.

Type pymongo.database.Database

is_valid (*self*)

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type bool

ping (*self*)

Pings the eve server and returns the result.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the JSON response from the server (probably "pong")

about (*self*)

Returns the ‘about’ dict from the server.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the JSON response from the server

Return type dict

get_resource (*self*, *resource*: str, *resource_id*: str)

Gets a resource from eve by its ID.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the JSON object response from the server

Return type dict

_add_or_replace_resource (*self*, *resource*: str, *obj*: dict, *valid_fn*: typing.Callable[[dict, typing.Callable[[str], None]], bool] = None)

Adds or replaces the given resource.

Parameters

- **resource** (*str*) – the resource type, e.g. "users"
- **obj** (*dict*) – the resource object
- **valid_fn** (*function*, *optional*) – a function to validate the resource object, defaults to None

Raises

- `exceptions.PineClientValueException` – if a `valid_fn` is passed in and the object fails
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the ID of the added/replaced resource

Return type `str`

`_add_resources` (*self*, *resource*: `str`, *objs*: `typing.List[dict]`, *valid_fn*: `typing.Callable[[dict, typing.Callable[[str], None]], bool] = None`, *replace_if_exists*: `bool = False`)

Tries to add all the resource objects at once, optionally falling back to individual replacement if that fails.

Parameters

- **resource** (`str`) – the resource type, e.g. "users"
- **objs** (`list(dict)`) – the resource objects
- **valid_fn** (`function, optional`) – a function to validate the resource object, defaults to `None`
- **replace_if_exists** (`bool, optional`) – whether to replace the resource with the given value if it already exists on the server, defaults to `False`

Raises

- `exceptions.PineClientValueException` – if a `valid_fn` is passed in and any of the objects fails
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the IDs of the added resources

Return type `list(str)`

`add_users` (*self*, *users*: `typing.List[dict]`, *replace_if_exists*=`False`)

Adds the given users.

Parameters

- **users** (`list(dict)`) – the user objects
- **replace_if_exists** (`bool, optional`) – whether to replace the resource with the given value if it already exists on the server, defaults to `False`

Raises

- `exceptions.PineClientValueException` – if any of the user objects are not valid, see `models.is_valid_eve_user()`
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the IDs of the added users

Return type `list(str)`

`get_users` (*self*)

Gets all users.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns all the users

Return type `list(dict)`

add_pipelines (*self*, *pipelines*: *typing.List[dict]*, *replace_if_exists=False*)

Adds the given pipelines.

Parameters

- **pipelines** (*list(dict)*) – the pipeline objects
- **replace_if_exists** (*bool*, *optional*) – whether to replace the resource with the given value if it already exists on the server, defaults to `False`

Raises

- ***exceptions.PineClientValueException*** – if any of the pipeline objects are not valid, see *models.is_valid_eve_pipeline()*
- ***exceptions.PineClientHttpException*** – if the HTTP request returns an error

Returns the IDs of the added pipelines

Return type *list(str)*

class *pine.client.client.PineClient* (*backend_base_uri: str*)

Bases: *pine.client.client.BaseClient*

A client to access PINE (more specifically: the backend).

Constructor.

Parameters **backend_base_uri** (*str*) – the base URI for the backend server, e.g. "http://localhost:5000"

is_valid (*self*)

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type *bool*

ping (*self*)

Pings the backend server and returns the result.

Raises ***exceptions.PineClientHttpException*** – if the HTTP request returns an error

Returns the JSON response from the server (probably "pong")

about (*self*)

Returns the 'about' dict from the server.

Raises ***exceptions.PineClientHttpException*** – if the HTTP request returns an error

Returns the JSON response from the server

Return type *dict*

get_logged_in_user (*self*)

Returns the currently logged in user, or None if not logged in.

Raises ***exceptions.PineClientHttpException*** – if the HTTP request returns an error

Returns currently logged in user, or None if not logged in

Return type *dict*

get_my_user_id (*self*)

Returns the ID of the logged in user, or None if not logged in.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the ID of the logged in user, or None if not logged in

Return type `str`

is_logged_in (*self*)

Returns whether the user is currently logged in or not.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns whether the user is currently logged in or not

Return type `bool`

_check_login (*self*)

Checks whether user is logged in and raises an `exceptions.PineClientAuthException` if not.

Raises

- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

get_auth_module (*self*)

Returns the PINE authentication module, e.g. "eve".

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the PINE authentication module, e.g. "eve"

Return type `str`

login_eve (*self*, *username: str*, *password: str*)

Logs in using eve credentials, and returns whether it was successful.

Parameters

- **username** (*str*) – username
- **password** (*str*) – password

Raises

- `exceptions.PineClientAuthException` – if auth module is not eve or login was not successful
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns whether the login was successful

Return type `bool`

logout (*self*)

Logs out the current user.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

get_pipelines (*self*)

Returns all pipelines accessible to logged in user.

Raises

- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns all pipelines accessible to logged in user

Return type `list(dict)`

collection_builder (*self*, ****kwargs**: *dict*)

Makes and returns a new `models.CollectionBuilder` with the logged in user.

Parameters ****kwargs** (*dict*) – any additional args to pass in to the constructor

Returns a new `models.CollectionBuilder` with the logged in user

Return type `models.CollectionBuilder`

create_collection (*self*, **builder**: `models.CollectionBuilder`)

Creates a collection using the current value of the given builder and returns its ID.

Parameters **builder** (`models.CollectionBuilder`) – collection builder

Raises

- `exceptions.PineClientValueException` – if the given collection is not valid, see `models.is_valid_collection()`
- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the created collection's ID

Return type `str`

get_collection_documents (*self*, **collection_id**: *str*, **truncate**: *bool*, **truncate_length**: *int* = 30)

Returns the documents in the given collection.

Parameters

- **collection_id** (*str*) – the ID of the collection
- **truncate** (*bool*) – whether to truncate the document text (a good idea unless you need it)
- **truncate_length** (*int*, *optional*) – how many characters of the text you want if truncated, defaults to 30

Returns the documents in the given collection

Return type `list(dict)`

add_document (*self*, **document**: *dict* = {}, **creator_id**: *str* = None, **collection_id**: *str* = None, **overlap**: *int* = None, **text**: *str* = None, **metadata**: *dict* = None)

Adds a new document to a collection and returns its ID.

Will use the logged in user ID for the `creator_id` if none is given. Although all the parameters are optional, you must provide values either in the document or through the kwargs in order to make a valid document.

Parameters

- **document** (*dict*, *optional*) – optional document dict, will be overridden with any kwargs, defaults to {}
- **creator_id** (*str*, *optional*) – optional `creator_id` for the document, defaults to None (not set)

- **collection_id** (*str*, *optional*) – optional collection_id for the document, defaults to None (not set)
- **overlap** (*int*, *optional*) – optional overlap for the document, defaults to None (not set)
- **text** (*str*, *optional*) – optional text for the document, defaults to None (not set)
- **metadata** (*dict*, *optional*) – optional metadata for the document, defaults to None (not set)

Raises

- **exceptions.PineClientValueException** – if the given document parameters are not valid, see `models.is_valid_eve_document()`
- **exceptions.PineClientAuthException** – if not logged in
- **exceptions.PineClientHttpException** – if the HTTP request returns an error

Returns the created document's ID

Return type *str*

add_documents (*self*, *documents*: *typing.List[dict]*, *creator_id*: *str* = None, *collection_id*: *str* = None)

Adds multiple documents at once and returns their IDs.

Will use the logged in user ID for the creator_id if none is given.

Parameters

- **documents** (*list(dict)*) – the documents to add
- **creator_id** (*str*, *optional*) – optional creator_id to set in the documents, defaults to None (not set)
- **collection_id** (*str*, *optional*) – optional collection_id to set in the documents, defaults to None (not set)

Raises

- **exceptions.PineClientValueException** – if any of the given documents are not valid, see `models.is_valid_eve_document()`
- **exceptions.PineClientAuthException** – if not logged in
- **exceptions.PineClientHttpException** – if the HTTP request returns an error

Returns the created documents' IDs

Return type *list(str)*

annotate_document (*self*, *document_id*: *str*, *doc_annotations*: *typing.List[str]*, *ner_annotations*: *typing.List[typing.Union[dict, list, tuple]]*)

Annotates the given document with the given values.

Parameters

- **document_id** (*str*) – the document ID to annotate
- **doc_annotations** (*list(str)*) – document annotations/labels
- **ner_annotations** (*list*) – NER annotations, where each annotation is either a list or a dict

Raises

- **`exceptions.PineClientValueException`** – if any of the given annotations are not valid, see `models.is_valid_annotation()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the annotation ID

Return type `str`

`annotate_collection_documents` (*self*, *collection_id*: `str`, *document_annotations*: `dict`, *skip_document_updates*=`False`)

Annotates documents in a collection.

Parameters

- **`collection_id`** (`str`) – the ID of the collection containing the documents to annotate
- **`document_annotations`** (`dict`) – a dict containing “ner” list and “doc” list
- **`skip_document_updates`** (`bool`) – whether to skip updating the document “has_annotated” map, defaults to `False`. This should only be `True` if you properly set the “has_annotated” map when you created the document.

Raises

- **`exceptions.PineClientValueException`** – if any of the given annotations are not valid, see `models.is_valid_doc_annotations()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the annotation IDs

Return type `list(str)`

`list_collections` (*self*, *include_archived*: `bool` = `False`)

Returns a list of user’s collections.

Parameters **`include_archived`** (`bool`) – whether to include archived collections, defaults to `False`

Raises

- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns user’s collections

Return type `list(dict)`

`download_collection_data` (*self*, *collection_id*: `str`, *include_collection_metadata*: `bool` = `True`, *include_document_metadata*: `bool` = `True`, *include_document_text*: `bool` = `True`, *include_annotations*: `bool` = `True`, *include_annotation_latest_version_only*: `bool` = `True`)

Downloads collection data.

Parameters

- **`collection_id`** (`str`) – the ID of the collection for which to download data
- **`include_collection_metadata`** (`bool`) – whether to include collection meta-data, defaults to `True`

- **include_document_metadata** (*bool*) – whether to include document metadata, defaults to `True`
- **include_document_text** (*bool*) – whether to include document text, defaults to `True`
- **include_annotations** (*bool*) – whether to include annotations, defaults to `True`
- **include_annotation_latest_version_only** (*bool*) – whether to include only the latest version of annotations (`True`) or all versions (`False`), defaults to `True`

Raises

- **`exceptions.PineClientValueException`** – if given empty collection ID
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error, such as if the collection doesn't exist

Returns collection data

Return type `dict`

```
class pine.client.client.LocalPineClient(backend_base_uri: str, eve_base_uri: str,
                                         mongo_base_uri: str = None, mongo_dbname: str = EveClient.DEFAULT_DBNAME)
```

Bases: `pine.client.client.PineClient`

A client for a local PINE instance, including an `:py:class<EveClient>`.

Constructor.

Parameters

- **backend_base_uri** (*str*) – the base URI for the backend server, e.g. `"http://localhost:5000"`
- **eve_base_uri** (*str*) – the base URI for the eve server, e.g. `"http://localhost:5001"`
- **mongo_base_uri** (*str*, *optional*) – the base URI for the mongodb server, e.g. `"mongodb://localhost:27018"`, defaults to `None`
- **mongo_dbname** (*str*, *optional*) – the DB name that PINE uses, defaults to `"pmap_nlp"`

eve :`EveClient`

The local `EveClient` instance.

Type `EveClient`

is_valid (*self*)

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type `bool`

`pine.client.exceptions`

PINE client exceptions module.

Module Contents

exception `pine.client.exceptions.PineClientException` (*message: str, cause: Exception = None*)

Bases: `Exception`

Base class for PINE client exceptions.

Constructor.

Parameters

- **message** (*str*) – the message
- **cause** (*Exception, optional*) – optional cause, defaults to None

message

The message.

Type `str`

exception `pine.client.exceptions.PineClientHttpException` (*method: str, path: str, resp: requests.Response*)

Bases: `pine.client.exceptions.PineClientException`

A PINE client exception caused by an underlying HTTP exception.

Constructor.

Parameters

- **method** (*str*) – the REST method ("get", "post", etc.)
- **path** (*str*) – the human-readable path that caused the exception
- **resp** (*requests.Response*) – the `Response` with the error info

resp

The `Response` with the error info

Type `requests.Response`

exception `pine.client.exceptions.PineClientValueException` (*obj: dict, obj_type: str*)

Bases: `pine.client.exceptions.PineClientException`

A PINE client exception caused by passing invalid data.

Constructor.

Parameters

- **obj** (*dict*) – the error data
- **obj_type** (*str*) – human-readable type of object

exception `pine.client.exceptions.PineClientAuthException` (*message: str, cause: Exception = None*)

Bases: `pine.client.exceptions.PineClientException`

Base class for PINE client exceptions.

Constructor.

Parameters

- **message** (*str*) – the message
- **cause** (*Exception*, *optional*) – optional cause, defaults to None

`pine.client.log`

Module Contents

Functions

<code>setup_logging()</code>	Sets up logging, if configured to do so.
------------------------------	--

`pine.client.log.CONFIG_FILE_ENV :str = PINE_LOGGING_CONFIG_FILE`
The environment variable that optionally contains the file to use for logging configuration.

Type `str`

`pine.client.log.setup_logging()`
Sets up logging, if configured to do so.

The environment variable named by `CONFIG_FILE_ENV` is checked and, if present, is passed to `logging.config.dictConfig()`.

`pine.client.models`

Module Contents

Classes

<code>CollectionBuilder(collection: dict = None, creator_id: str = None, viewers: typing.List[str] = None, annotators: typing.List[str] = None, labels: typing.List[str] = None, title: str = None, description: str = None, allow_overlapping_ner_annotations: bool = None, pipelineId: str = None, overlap: float = None, train_every: int = None, classifierParameters: dict = None, document_csv_file: str = None, document_csv_file_has_header: bool = None, document_csv_file_text_column: int = None, image_files: typing.List[str] = None)</code>	A class that can build the form and files fields that are necessary to create a collection.
---	---

Functions

<code>_check_field_required_bool(obj: dict, field: str)</code>	Checks that the given field is in the object and is a bool.
<code>_check_field_int(obj: dict, field: str)</code>	Checks that if the given field is in the object, that it is an int.
<code>_check_field_required_int(obj: dict, field: str)</code>	Checks that the given field is in the object and is an int.
<code>_check_field_float(obj: dict, field: str)</code>	Checks that if the given field is in the object, that it is a float.
<code>_check_field_required_float(obj: dict, field: str)</code>	Checks that the given field is in the object and is a float.
<code>_check_field_string(obj: dict, field: str)</code>	Checks that if the given field is in the object, that it is a string.
<code>_check_field_required_string(obj: dict, field: str)</code>	Checks that the given field is in the object and is a string.
<code>_check_field_string_list(obj: dict, field: str, min_length: int = 0)</code>	Checks that if the given field is in the object, that it is a string list.
<code>_check_field_required_string_list(obj: dict, field: str, min_length: int = 0)</code>	Checks that the given field is in the object and is a string list.
<code>_check_field_dict(obj: dict, field: str)</code>	Checks that if the given field is in the object, that it is a dict.
<code>_check_field_required_dict(obj: dict, field: str)</code>	Checks that the given field is in the object and is a dict.
<code>_check_field_bool(obj: dict, field: str)</code>	Checks that if the given field is in the object, that it is a bool.
<code>is_valid_eve_user(user: dict, error_callback: typing.Callable[[str], None] = None)</code>	Checks whether the given user object is valid.
<code>is_valid_eve_pipeline(pipeline: dict, error_callback: typing.Callable[[str], None] = None)</code>	Checks whether the given pipeline object is valid.
<code>is_valid_eve_collection(collection: dict, error_callback: typing.Callable[[str], None] = None)</code>	Checks whether the given collection object is valid.
<code>is_valid_collection(form: dict, files: dict, error_callback: typing.Callable[[str], None] = None)</code>	Checks whether the given form and files parameters are valid for creating a collection.
<code>is_valid_eve_document(document: dict, error_callback: typing.Callable[[str], None] = None)</code>	Checks whether the given document object is valid.
<code>is_valid_doc_annotation(ann: typing.Any, error_callback: typing.Callable[[str], None] = None)</code>	Checks whether the given annotation is a valid document label/annotation.
<code>is_valid_ner_annotation(ann: typing.Any, error_callback: typing.Callable[[str], None] = None)</code>	Checks whether the given annotation is a valid document NER annotation.
<code>is_valid_annotation(body: dict, error_callback: typing.Callable[[str], None] = None)</code>	Checks whether the given body is valid to create an annotation.
<code>is_valid_doc_annotations(doc_annotations: dict, error_callback: typing.Callable[[str], None] = None)</code>	Checks whether the given document annotations are valid.
<code>remove_eve_fields(obj: dict, remove_timestamps: bool = True, remove_versions: bool = True)</code>	Removes fields inserted by eve from the given object.
<code>remove_nonupdatable_fields(obj: dict)</code>	Removes all non-updatable fields from the given object.

`pine.client.models.ID_FIELD :str = _id`

The field used to store database ID.

Type str

`pine.client.models.ITEMS_FIELD :str = _items`

The field used to access the items in a multi-item database response.

Type str

`pine.client.models._check_field_required_bool (obj: dict, field: str) → bool`
Checks that the given field is in the object and is a bool.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns whether the given field is in the object and is a bool

Return type `bool`

`pine.client.models._check_field_int (obj: dict, field: str) → bool`
Checks that if the given field is in the object, that it is an int.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns if the given field is in the object, that it is an int

Return type `bool`

`pine.client.models._check_field_required_int (obj: dict, field: str) → bool`
Checks that the given field is in the object and is an int.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns whether the given field is in the object and is an int

Return type `bool`

`pine.client.models._check_field_float (obj: dict, field: str) → bool`
Checks that if the given field is in the object, that it is a float.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns if the given field is in the object, that it is a float

Return type `bool`

`pine.client.models._check_field_required_float (obj: dict, field: str) → bool`
Checks that the given field is in the object and is a float.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns whether the given field is in the object and is a float

Return type `bool`

`pine.client.models._check_field_string (obj: dict, field: str) → bool`
Checks that if the given field is in the object, that it is a string.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns if the given field is in the object, that it is a string

Return type `bool`

`pine.client.models._check_field_required_string(obj: dict, field: str) → bool`

Checks that the given field is in the object and is a string.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns whether the given field is in the object and is a string

Return type `bool`

`pine.client.models._check_field_string_list(obj: dict, field: str, min_length: int = 0) →`

Checks that if the given field is in the object, that it is a string ^{`bool`} list.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check
- **min_length** (*int*, *optional*) – the minimum length of the list (if > 0), defaults to 0

Returns if the given field is in the object, that it is a string list

Return type `bool`

`pine.client.models._check_field_required_string_list(obj: dict, field: str, min_length: int = 0) → bool`

Checks that the given field is in the object and is a string list.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check
- **min_length** (*int*, *optional*) – the minimum length of the list (if > 0), defaults to 0

Returns if the given field is in the object, that it is a string list

Return type `bool`

`pine.client.models._check_field_dict(obj: dict, field: str) → bool`

Checks that if the given field is in the object, that it is a dict.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns if the given field is in the object, that it is a dict

Return type `bool`

`pine.client.models._check_field_required_dict(obj: dict, field: str) → bool`

Checks that the given field is in the object and is a dict.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns whether the given field is in the object and is a dict

Return type `bool`

```
pine.client.models._check_field_bool (obj: dict, field: str) → bool
```

Checks that if the given field is in the object, that it is a bool.

Parameters

- **obj** (*dict*) – the object to check
- **field** (*str*) – the field to check

Returns if the given field is in the object, that it is a bool

Return type `bool`

```
pine.client.models.is_valid_eve_user (user: dict, error_callback: typing.Callable[[str], None]
                                     = None) → bool
```

Checks whether the given user object is valid.

A valid user object has an `_id`, `firstname`, and `lastname` that are non-empty string fields. If `email`, `description`, or `passwdhash` are present, they are string fields. If `role` is present, it is a list of strings that are either `administrator` or `user`.

Parameters

- **user** (*dict*) – user object
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to `None`

Returns whether the given user object is valid

Return type `bool`

```
pine.client.models.is_valid_eve_pipeline (pipeline: dict, error_callback: typing.Callable[[str], None] = None) → bool
```

Checks whether the given pipeline object is valid.

A valid pipeline has an `_id`, `title`, and `name` that are non-empty string fields. If `description` is provided, it is a string field. If `parameters` are provided, it is a dict field.

Parameters

- **pipeline** (*dict*) – pipeline object
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to `None`

Returns whether the given pipeline object is valid

Return type `bool`

```
pine.client.models.is_valid_eve_collection (collection: dict, error_callback: typing.Callable[[str], None] = None) → bool
```

Checks whether the given collection object is valid.

A valid collection has a `creator_id` that is a non-empty string field. It has a `labels` that is a non-empty list of strings. If `annotators` or `viewers` are provided, they are lists of strings. If `metadata` or `configuration` are provided, they are dicts. If `archived` is provided, it is a bool.

Parameters

- **collection** (*dict*) – collection object
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given collection object is valid

Return type `bool`

```
pine.client.models.is_valid_collection (form: dict, files: dict, error_callback: typing.Callable[[str], None] = None) → bool
```

Checks whether the given form and files parameters are valid for creating a collection.

A valid form has a `collection` that is a dict field and is valid via `is_valid_eve_collection()`. Additionally, the collection has string `title` and `description` fields in its metadata. It also has at least one element for `labels`, `viewers`, and `annotators`, and the `creator_id` must be in both `viewers` and `annotators`.

The form also has `overlap` as a float field between 0 and 1 (inclusive), `train_every` as an int field that is at least 5, and `pipelineId` as a string field.

If files are provided, file `file` and any files starting with `imageFile` are checked. If a file `file` is provided, the form must also have a boolean `csvHasHeader` field and an int `csvTextCol` field.

Parameters

- **form** (*dict*) – form data to send to backend
- **files** (*dict*) – file data to send to backend
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given form and files parameters are valid for creating a collection

Return type `bool`

```
pine.client.models.is_valid_eve_document (document: dict, error_callback: typing.Callable[[str], None] = None) → bool
```

Checks whether the given document object is valid.

A valid document has a `creator_id` and `collection_id` that are non-empty string fields. Optionally, it may have an int `overlap` field, string `text` field, and dict `metadata` and `has_annotated` fields.

Parameters

- **document** (*dict*) – document object
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given document object is valid

Return type `bool`

```
pine.client.models.is_valid_doc_annotation (ann: typing.Any, error_callback: typing.Callable[[str], None] = None) → bool
```

Checks whether the given annotation is a valid document label/annotation.

This means that it is a non-empty string.

Parameters

- **ann** – annotation

- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given annotation is a valid document label/annotation

Return type `bool`

```
pine.client.models.is_valid_ner_annotation(ann: typing.Any, error_callback: typing.Callable[[str], None] = None) → bool
```

Checks whether the given annotation is a valid document NER annotation.

Valid NER annotations take one of two forms: a `dict` or a `list/tuple` of size 3.

A valid NER `dict` has the following fields:

- `start`: an `int` that is ≥ 0
- `end`: an `int` that is ≥ 0
- `label`: a non-empty `str`

A valid NER `list/tuple` has the following elements:

- `element 0`: an `int` that is ≥ 0
- `element 1`: an `int` that is ≥ 0
- `element 2`: a non-empty `str`

Parameters

- **ann** – annotation
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given annotation is a valid document label/annotation

Return type `bool`

```
pine.client.models.is_valid_annotation(body: dict, error_callback: typing.Callable[[str], None] = None) → bool
```

Checks whether the given body is valid to create an annotation.

A valid body is a `dict` with two fields:

- `doc`: a list of valid doc annotations (see `is_valid_doc_annotation()`)
- `ner`: a list of valid NER annotations (see `is_valid_ner_annotation()`)

Parameters

- **body** (*dict*) – annotation body
- **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None

Returns whether the given body is valid to create an annotation

Return type `bool`

```
pine.client.models.is_valid_doc_annotations(doc_annotations: dict, error_callback: typing.Callable[[str], None] = None) → bool
```

Checks whether the given document annotations are valid.

A valid document annotations object is a `dict`, where the keys are `str` document IDs, and the values are valid annotation bodies (see `is_valid_annotation()`).

Parameters

- **doc_annotations** – document annotations
- **error_callback** (*function, optional*) – optional callback that is called with any error messages, defaults to `None`

Returns whether the given body is valid to create an annotation

Return type `bool`

```
class pine.client.models.CollectionBuilder (collection: dict = None, creator_id: str = None, viewers: typing.List[str] = None,
annotators: typing.List[str] = None, labels: typing.List[str] = None, title: str = None, description: str = None, allow_overlapping_ner_annotations: bool = None, pipelineId: str = None, overlap: float = None, train_every: int = None, classifierParameters: dict = None, document_csv_file: str = None, document_csv_file_has_header: bool = None, document_csv_file_text_column: int = None, image_files: typing.List[str] = None)
```

Bases: `object`

A class that can build the form and files fields that are necessary to create a collection.

Constructor.

Parameters

- **collection** (*dict, optional*) – starting parameters for the collection, defaults to `None` (not set)
- **creator_id** (*str, optional*) – user ID for the creator, see `creator_id()`, defaults to `None` (not set)
- **viewers** (*list(str), optional*) – viewer IDs for the collection, see `viewer()`, defaults to `None` (not set)
- **annotators** (*list(str), optional*) – annotator IDs for the collection, see `annotator()`, defaults to `None` (not set)
- **labels** (*list(str), optional*) – labels for the collection, see `label()`, defaults to `None` (not set)
- **title** (*str, optional*) – metadata title, see `title()`, defaults to `None` (not set)
- **description** (*str, optional*) – metadata description, see `description()`, defaults to `None` (not set)
- **allow_overlapping_ner_annotations** (*bool*) – optional configuration for allowing overlapping NER annotations, see `allow_overlapping_ner_annotations()`, defaults to `None` (not set)
- **pipelineId** (*str, optional*) – the ID of the pipeline from which to create the classifier, see `classifier()`, defaults to `None` (not set)
- **overlap** (*float, optional*) – the classifier overlap, see `classifier()`, defaults to `None` (not set)

- **train_every** (*int*, *optional*) – train the model after this many documents are annotated, see *classifier()*, defaults to None (not set)
- **classifierParameters** (*dict*, *optional*) – any parameters to pass to the classifier, see *classifier()*, defaults to None (not set)
- **document_csv_file** (*str*, *optional*) – the filename of the local document CSV file, see *document_csv_File()*, defaults to None (not set)
- **document_csv_file_has_header** (*bool*, *optional*) – whether the document CSV file has a header, see *document_csv_File()*, defaults to None (not set)
- **document_csv_file_text_column** (*int*, *optional*) – if the document CSV file has headers, the document text can be found in this column index (the others are used for document metadata), see *document_csv_File()*, defaults to None (not set)
- **image_files** (*list(str)*) – any image files to add to the collection, see *image_file()*, defaults to None (not set)

form

The form data.

Type *dict*

files

The files data.

Type *dict*

property collection (*self*)

Returns the collection information from the form.

Returns collection information from the form

Return type *dict*

property form_json (*self*)

Returns the form where the values have been JSON-encoded.

Returns the form where the values have been JSON-encoded

Return type *dict*

creator_id (*self*, *user_id: str*)

Sets the *creator_id* to the given, and adds to viewers and annotators.

Parameters **user_id** (*str*) – the user ID to use for the *creator_id*

Returns *self*

Return type *models.CollectionBuilder*

viewer (*self*, *user_id: str*)

Adds the given user to the list of viewers.

Parameters **user_id** (*str*) – the user ID to add as a viewer

Returns *self*

Return type *models.CollectionBuilder*

annotator (*self*, *user_id: str*)

Adds the given user to the list of annotators.

Parameters **user_id** (*str*) – the user ID to add as an annotator

Returns *self*

Return type *models.CollectionBuilder*

label (*self*, *label*: *str*)

Adds the given label to the collection.

Parameters **label** (*str*) – label to add

Returns *self*

Return type *models.CollectionBuilder*

metadata (*self*, *key*: *str*, *value*: *typing.Any*)

Adds the given metadata key/value to the collection.

Parameters

- **key** (*str*) – metadata key
- **value** – metadata value

Returns *self*

Return type *models.CollectionBuilder*

title (*self*, *title*: *str*)

Sets the metadata title to the given.

Parameters **title** (*str*) – collection title

Returns *self*

Return type *models.CollectionBuilder*

description (*self*, *description*: *str*)

Sets the metadata description to the given.

Parameters **description** (*str*) – collection description

Returns *self*

Return type *models.CollectionBuilder*

configuration (*self*, *key*: *str*, *value*: *typing.Any*)

Adds the given configuration key/value to the collection.

Parameters

- **key** (*str*) – configuration key
- **value** – configuration value

Returns *self*

Return type *models.CollectionBuilder*

allow_overlapping_ner_annotations (*self*, *allow_overlapping_ner_annotations*: *bool*)

Sets the configuration value for `allow_overlapping_ner_annotations` to the given.

Parameters **allow_overlapping_ner_annotations** (*bool*) – whether to allow overlapping NER annotations

Returns *self*

Return type *models.CollectionBuilder*

classifier (*self*, *pipelineId*: *str*, *overlap*: *float* = 0, *train_every*: *int* = 100, *classifierParameters*: *dict* = {})

Sets classifier information for the created collection.

Parameters

- **pipelineId** (*str*) – the ID of the pipeline from which to create the classifier
- **overlap** (*float*, *optional*) – the classifier overlap, defaults to 0
- **train_every** (*int*, *optional*) – train the model after this many documents are annotated, defaults to 100
- **classifierParameters** (*dict*, *optional*) – any parameters to pass to the classifier, defaults to { }

Returns self**Return type** *models.CollectionBuilder***document_csv_file** (*self*, *csv_filename: str*, *has_header: bool*, *text_column: int*)

Sets the CSV file used to create documents to the given.

May raise an Exception if there is a problem opening the indicated file.

Parameters

- **csv_filename** (*str*) – the filename of the local CSV file
- **has_header** (*bool*) – whether the CSV file has a header
- **text_column** (*int*) – if the CSV file has headers, the document text can be found in this column index (the others are used for document metadata)

Returns self**Return type** *models.CollectionBuilder***image_file** (*self*, *image_filename: str*)

Adds the given image file to the collection.

May raise an Exception if there is a problem opening the indicated file.

Parameters **image_filename** (*str*) – the filename of the local image file**Returns** self**Return type** *models.CollectionBuilder***is_valid** (*self*, *error_callback: typing.Callable[[str], None] = None*)

Checks whether the currently set values are valid or not.

See *is_valid_collection()*.**Parameters** **error_callback** (*function*, *optional*) – optional callback that is called with any error messages, defaults to None**Returns** whether the currently set values are valid or not**Return type** *bool*

`pine.client.models.remove_eve_fields(obj: dict, remove_timestamps: bool = True, remove_versions: bool = True)`

Removes fields inserted by eve from the given object.

Parameters

- **obj** (*dict*) – the object
- **remove_timestamps** (*bool*) – whether to remove the timestamp fields, defaults to True
- **remove_versions** (*bool*) – whether to remove the version fields, defaults to True

`pine.client.models.remove_nonupdatable_fields(obj: dict)`

Removes all non-updatable fields from the given object.

These fields would cause a PUT/PATCH to be rejected because they are not user-modifiable.

Parameters `obj` (*dict*) – the object

`pine.client.password`

Module Contents

Functions

<code>hash_password(password: str)</code>	Hashes the given password for use in user object.
<code>check_password(password: str, hashed_password: str)</code>	Checks the given password against the given hash.

`pine.client.password.hash_password(password: str) → str`

Hashes the given password for use in user object.

Parameters `password` (*str*) – password

Returns hashed password

Return type *str*

`pine.client.password.check_password(password: str, hashed_password: str) → str`

Checks the given password against the given hash.

Parameters

- `password` (*str*) – password to check
- `hashed_password` (*str*) – hashed password to check against

Returns whether the password matches the hash

Return type *bool*

Package Contents

Classes

<code>PineClient(backend_base_uri: str)</code>	A client to access PINE (more specifically: the backend).
<code>LocalPineClient(backend_base_uri: str, eve_base_uri: str, mongo_base_uri: str = None, mongo_dbname: str = EveClient.DEFAULT_DBNAME)</code>	A client for a local PINE instance, including an <code>:py:class<.EveClient></code> .

continues on next page

Table 43 – continued from previous page

<code>CollectionBuilder</code> (collection: dict = None, creator_id: str = None, viewers: typing.List[str] = None, annotators: typing.List[str] = None, labels: typing.List[str] = None, title: str = None, description: str = None, allow_overlapping_ner_annotations: bool = None, pipelineId: str = None, overlap: float = None, train_every: int = None, classifierParameters: dict = None, document_csv_file: str = None, document_csv_file_has_header: bool = None, document_csv_file_text_column: int = None, image_files: typing.List[str] = None)	A class that can build the form and files fields that are necessary to create a collection.
--	---

Functions

<code>setup_logging()</code>	Sets up logging, if configured to do so.
------------------------------	--

class `pine.client.PineClient` (*backend_base_uri: str*)

Bases: `pine.client.client.BaseClient`

A client to access PINE (more specifically: the backend).

Constructor.

Parameters `backend_base_uri` (*str*) – the base URI for the backend server, e.g. "http://localhost:5000"

is_valid (*self*)

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type `bool`

ping (*self*)

Pings the backend server and returns the result.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the JSON response from the server (probably "pong")

about (*self*)

Returns the 'about' dict from the server.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the JSON response from the server

Return type `dict`

get_logged_in_user (*self*)

Returns the currently logged in user, or None if not logged in.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns currently logged in user, or None if not logged in

Return type `dict`

get_my_user_id (*self*)

Returns the ID of the logged in user, or None if not logged in.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the ID of the logged in user, or None if not logged in

Return type `str`

is_logged_in (*self*)

Returns whether the user is currently logged in or not.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns whether the user is currently logged in or not

Return type `bool`

_check_login (*self*)

Checks whether user is logged in and raises an `exceptions.PineClientAuthException` if not.

Raises

- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

get_auth_module (*self*)

Returns the PINE authentication module, e.g. "eve".

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the PINE authentication module, e.g. "eve"

Return type `str`

login_eve (*self*, *username: str*, *password: str*)

Logs in using eve credentials, and returns whether it was successful.

Parameters

- **username** (*str*) – username
- **password** (*str*) – password

Raises

- `exceptions.PineClientAuthException` – if auth module is not eve or login was not successful
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns whether the login was successful

Return type `bool`

logout (*self*)

Logs out the current user.

Raises `exceptions.PineClientHttpException` – if the HTTP request returns an error

get_pipelines (*self*)

Returns all pipelines accessible to logged in user.

Raises

- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns all pipelines accessible to logged in user

Return type `list(dict)`

collection_builder (*self*, ****kwargs**: *dict*)

Makes and returns a new `models.CollectionBuilder` with the logged in user.

Parameters ****kwargs** (*dict*) – any additional args to pass in to the constructor

Returns a new `models.CollectionBuilder` with the logged in user

Return type `models.CollectionBuilder`

create_collection (*self*, **builder**: `models.CollectionBuilder`)

Creates a collection using the current value of the given builder and returns its ID.

Parameters **builder** (`models.CollectionBuilder`) – collection builder

Raises

- `exceptions.PineClientValueException` – if the given collection is not valid, see `models.is_valid_collection()`
- `exceptions.PineClientAuthException` – if not logged in
- `exceptions.PineClientHttpException` – if the HTTP request returns an error

Returns the created collection's ID

Return type `str`

get_collection_documents (*self*, **collection_id**: *str*, **truncate**: *bool*, **truncate_length**: *int* = 30)

Returns the documents in the given collection.

Parameters

- **collection_id** (*str*) – the ID of the collection
- **truncate** (*bool*) – whether to truncate the document text (a good idea unless you need it)
- **truncate_length** (*int*, *optional*) – how many characters of the text you want if truncated, defaults to 30

Returns the documents in the given collection

Return type `list(dict)`

add_document (*self*, **document**: *dict* = {}, **creator_id**: *str* = None, **collection_id**: *str* = None, **overlap**: *int* = None, **text**: *str* = None, **metadata**: *dict* = None)

Adds a new document to a collection and returns its ID.

Will use the logged in user ID for the `creator_id` if none is given. Although all the parameters are optional, you must provide values either in the document or through the kwargs in order to make a valid document.

Parameters

- **document** (*dict*, *optional*) – optional document dict, will be overridden with any kwargs, defaults to {}

- **creator_id** (*str*, *optional*) – optional creator_id for the document, defaults to None (not set)
- **collection_id** (*str*, *optional*) – optional collection_id for the document, defaults to None (not set)
- **overlap** (*int*, *optional*) – optional overlap for the document, defaults to None (not set)
- **text** (*str*, *optional*) – optional text for the document, defaults to None (not set)
- **metadata** (*dict*, *optional*) – optional metadata for the document, defaults to None (not set)

Raises

- **exceptions.PineClientValueException** – if the given document parameters are not valid, see `models.is_valid_eve_document()`
- **exceptions.PineClientAuthException** – if not logged in
- **exceptions.PineClientHttpException** – if the HTTP request returns an error

Returns the created document's ID

Return type `str`

add_documents (*self*, *documents*: `typing.List[dict]`, *creator_id*: `str = None`, *collection_id*: `str = None`)
Adds multiple documents at once and returns their IDs.

Will use the logged in user ID for the creator_id if none is given.

Parameters

- **documents** (`list(dict)`) – the documents to add
- **creator_id** (*str*, *optional*) – optional creator_id to set in the documents, defaults to None (not set)
- **collection_id** (*str*, *optional*) – optional collection_id to set in the documents, defaults to None (not set)

Raises

- **exceptions.PineClientValueException** – if any of the given documents are not valid, see `models.is_valid_eve_document()`
- **exceptions.PineClientAuthException** – if not logged in
- **exceptions.PineClientHttpException** – if the HTTP request returns an error

Returns the created documents' IDs

Return type `list(str)`

annotate_document (*self*, *document_id*: *str*, *doc_annotations*: `typing.List[str]`, *ner_annotations*: `typing.List[typing.Union[dict, list, tuple]]`)
Annotates the given document with the given values.

Parameters

- **document_id** (*str*) – the document ID to annotate
- **doc_annotations** (`list(str)`) – document annotations/labels
- **ner_annotations** (`list`) – NER annotations, where each annotation is either a list or a dict

Raises

- **`exceptions.PineClientValueException`** – if any of the given annotations are not valid, see `models.is_valid_annotation()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the annotation ID

Return type `str`

`annotate_collection_documents` (*self*, *collection_id*: `str`, *document_annotations*: `dict`, *skip_document_updates*=`False`)

Annotates documents in a collection.

Parameters

- **`collection_id`** (`str`) – the ID of the collection containing the documents to annotate
- **`document_annotations`** (`dict`) – a dict containing “ner” list and “doc” list
- **`skip_document_updates`** (`bool`) – whether to skip updating the document “has_annotated” map, defaults to `False`. This should only be `True` if you properly set the “has_annotated” map when you created the document.

Raises

- **`exceptions.PineClientValueException`** – if any of the given annotations are not valid, see `models.is_valid_doc_annotations()`
- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns the annotation IDs

Return type `list(str)`

`list_collections` (*self*, *include_archived*: `bool` = `False`)

Returns a list of user’s collections.

Parameters **`include_archived`** (`bool`) – whether to include archived collections, defaults to `False`

Raises

- **`exceptions.PineClientAuthException`** – if not logged in
- **`exceptions.PineClientHttpException`** – if the HTTP request returns an error

Returns user’s collections

Return type `list(dict)`

`download_collection_data` (*self*, *collection_id*: `str`, *include_collection_metadata*: `bool` = `True`, *include_document_metadata*: `bool` = `True`, *include_document_text*: `bool` = `True`, *include_annotations*: `bool` = `True`, *include_annotation_latest_version_only*: `bool` = `True`)

Downloads collection data.

Parameters

- **`collection_id`** (`str`) – the ID of the collection for which to download data
- **`include_collection_metadata`** (`bool`) – whether to include collection meta-data, defaults to `True`

- **include_document_metadata** (*bool*) – whether to include document metadata, defaults to True
- **include_document_text** (*bool*) – whether to include document text, defaults to True
- **include_annotations** (*bool*) – whether to include annotations, defaults to True
- **include_annotation_latest_version_only** (*bool*) – whether to include only the latest version of annotations (True) or all versions (False), defaults to True

Raises

- **exceptions.PineClientValueException** – if given empty collection ID
- **exceptions.PineClientAuthException** – if not logged in
- **exceptions.PineClientHttpException** – if the HTTP request returns an error, such as if the collection doesn't exist

Returns collection data

Return type *dict*

```
class pine.client.LocalPineClient (backend_base_uri: str, eve_base_uri: str,
                                  mongo_base_uri: str = None, mongo_dbname: str =
                                  EveClient.DEFAULT_DBNAME)
```

Bases: *pine.client.client.PineClient*

A client for a local PINE instance, including an :py:class<EveClient>.

Constructor.

Parameters

- **backend_base_uri** (*str*) – the base URI for the backend server, e.g. "http://localhost:5000"
- **eve_base_uri** (*str*) – the base URI for the eve server, e.g. "http://localhost:5001"
- **mongo_base_uri** (*str*, *optional*) – the base URI for the mongodb server, e.g. "mongodb://localhost:27018", defaults to None
- **mongo_dbname** (*str*, *optional*) – the DB name that PINE uses, defaults to "pmap_nlp"

eve :EveClient

The local EveClient instance.

Type *EveClient*

is_valid (*self*)

Returns whether this client and its connection(s) are valid.

Returns whether this client and its connection(s) are valid

Return type *bool*

```
pine.client.setup_logging()
```

Sets up logging, if configured to do so.

The environment variable named by CONFIG_FILE_ENV is checked and, if present, is passed to `logging.config.dictConfig()`.

```
class pine.client.CollectionBuilder(collection: dict = None, creator_id: str = None,
                                   viewers: typing.List[str] = None, annotators: typing.List[str] = None, labels: typing.List[str] = None, title: str = None, description: str = None,
                                   allow_overlapping_ner_annotations: bool = None, pipelineId: str = None, overlap: float = None, train_every: int = None, classifierParameters: dict = None, document_csv_file: str = None, document_csv_file_has_header: bool = None, document_csv_file_text_column: int = None, image_files: typing.List[str] = None)
```

Bases: `object`

A class that can build the form and files fields that are necessary to create a collection.

Constructor.

Parameters

- **collection** (*dict*, *optional*) – starting parameters for the collection, defaults to `None` (not set)
- **creator_id** (*str*, *optional*) – user ID for the creator, see `creator_id()`, defaults to `None` (not set)
- **viewers** (*list(str)*, *optional*) – viewer IDs for the collection, see `viewer()`, defaults to `None` (not set)
- **annotators** (*list(str)*, *optional*) – annotator IDs for the collection, see `annotator()`, defaults to `None` (not set)
- **labels** (*list(str)*, *optional*) – labels for the collection, see `label()`, defaults to `None` (not set)
- **title** (*str*, *optional*) – metadata title, see `title()`, defaults to `None` (not set)
- **description** (*str*, *optional*) – metadata description, see `description()`, defaults to `None` (not set)
- **allow_overlapping_ner_annotations** (*bool*) – optional configuration for allowing overlapping NER annotations, see `allow_overlapping_ner_annotations()`, defaults to `None` (not set)
- **pipelineId** (*str*, *optional*) – the ID of the pipeline from which to create the classifier, see `classifier()`, defaults to `None` (not set)
- **overlap** (*float*, *optional*) – the classifier overlap, see `classifier()`, defaults to `None` (not set)
- **train_every** (*int*, *optional*) – train the model after this many documents are annotated, see `classifier()`, defaults to `None` (not set)
- **classifierParameters** (*dict*, *optional*) – any parameters to pass to the classifier, see `classifier()`, defaults to `None` (not set)
- **document_csv_file** (*str*, *optional*) – the filename of the local document CSV file, see `document_csv_File()`, defaults to `None` (not set)
- **document_csv_file_has_header** (*bool*, *optional*) – whether the document CSV file has a header, see `document_csv_File()`, defaults to `None` (not set)

- **document_csv_file_text_column** (*int*, *optional*) – if the document CSV file has headers, the document text can be found in this column index (the others are used for document metadata), see `document_csv_File()`, defaults to `None` (not set)
- **image_files** (*list(str)*) – any image files to add to the collection, see `image_file()`, defaults to `None` (not set)

form

The form data.

Type `dict`

files

The files data.

Type `dict`

property collection (*self*)

Returns the collection information from the form.

Returns collection information from the form

Return type `dict`

property form_json (*self*)

Returns the form where the values have been JSON-encoded.

Returns the form where the values have been JSON-encoded

Return type `dict`

creator_id (*self*, *user_id: str*)

Sets the `creator_id` to the given, and adds to viewers and annotators.

Parameters **user_id** (*str*) – the user ID to use for the `creator_id`

Returns `self`

Return type `models.CollectionBuilder`

viewer (*self*, *user_id: str*)

Adds the given user to the list of viewers.

Parameters **user_id** (*str*) – the user ID to add as a viewer

Returns `self`

Return type `models.CollectionBuilder`

annotator (*self*, *user_id: str*)

Adds the given user to the list of annotators.

Parameters **user_id** (*str*) – the user ID to add as an annotator

Returns `self`

Return type `models.CollectionBuilder`

label (*self*, *label: str*)

Adds the given label to the collection.

Parameters **label** (*str*) – label to add

Returns `self`

Return type `models.CollectionBuilder`

metadata (*self*, *key*: *str*, *value*: *typing.Any*)

Adds the given metadata key/value to the collection.

Parameters

- **key** (*str*) – metadata key
- **value** – metadata value

Returns *self*

Return type *models.CollectionBuilder*

title (*self*, *title*: *str*)

Sets the metadata title to the given.

Parameters **title** (*str*) – collection title

Returns *self*

Return type *models.CollectionBuilder*

description (*self*, *description*: *str*)

Sets the metadata description to the given.

Parameters **description** (*str*) – collection description

Returns *self*

Return type *models.CollectionBuilder*

configuration (*self*, *key*: *str*, *value*: *typing.Any*)

Adds the given configuration key/value to the collection.

Parameters

- **key** (*str*) – configuration key
- **value** – configuration value

Returns *self*

Return type *models.CollectionBuilder*

allow_overlapping_ner_annotations (*self*, *allow_overlapping_ner_annotations*: *bool*)

Sets the configuration value for `allow_overlapping_ner_annotations` to the given.

Parameters **allow_overlapping_ner_annotations** (*bool*) – whether to allow overlapping NER annotations

Returns *self*

Return type *models.CollectionBuilder*

classifier (*self*, *pipelineId*: *str*, *overlap*: *float* = 0, *train_every*: *int* = 100, *classifierParameters*: *dict* = {})

Sets classifier information for the created collection.

Parameters

- **pipelineId** (*str*) – the ID of the pipeline from which to create the classifier
- **overlap** (*float*, *optional*) – the classifier overlap, defaults to 0
- **train_every** (*int*, *optional*) – train the model after this many documents are annotated, defaults to 100
- **classifierParameters** (*dict*, *optional*) – any parameters to pass to the classifier, defaults to { }

Returns `self`

Return type `models.CollectionBuilder`

document_csv_file (*self*, *csv_filename*: `str`, *has_header*: `bool`, *text_column*: `int`)

Sets the CSV file used to create documents to the given.

May raise an Exception if there is a problem opening the indicated file.

Parameters

- **csv_filename** (`str`) – the filename of the local CSV file
- **has_header** (`bool`) – whether the CSV file has a header
- **text_column** (`int`) – if the CSV file has headers, the document text can be found in this column index (the others are used for document metadata)

Returns `self`

Return type `models.CollectionBuilder`

image_file (*self*, *image_filename*: `str`)

Adds the given image file to the collection.

May raise an Exception if there is a problem opening the indicated file.

Parameters **image_filename** (`str`) – the filename of the local image file

Returns `self`

Return type `models.CollectionBuilder`

is_valid (*self*, *error_callback*: `typing.Callable[[str], None] = None`)

Checks whether the currently set values are valid or not.

See `is_valid_collection()`.

Parameters **error_callback** (`function`, `optional`) – optional callback that is called with any error messages, defaults to None

Returns whether the currently set values are valid or not

Return type `bool`

`pine.pipelines`

Subpackages

`pine.pipelines.app`

Subpackages

`pine.pipelines.app.listener`

Submodules

`pine.pipelines.app.listener.main`

Module Contents

Functions

`backoff_hdlr(details)`

`setup_logging()`

`run()`

`pine.pipelines.app.listener.main.backoff_hdlr(details)``pine.pipelines.app.listener.main.setup_logging()``pine.pipelines.app.listener.main.run()``pine.pipelines.app.listener.service_listener`

Module Contents

Classes

`ServiceRegistration(name, version, channel, Initialize self. See help(type(self)) for accurate signature.
framework, framework_types)`

`ServiceListener(services=None)``type services list[ServiceRegistration]`

`pine.pipelines.app.listener.service_listener.config``pine.pipelines.app.listener.service_listener.logger`

```
class pine.pipelines.app.listener.service_listener.ServiceRegistration(name,  
                                                                      ver-  
                                                                      sion,  
                                                                      chan-  
                                                                      nel,  
                                                                      frame-  
                                                                      work,  
                                                                      frame-  
                                                                      work_types)
```

`Bases: object``Initialize self. See help(type(self)) for accurate signature.``classmethod from_registration_format(cls, **kwargs)``to_registration_format(self)``class pine.pipelines.app.listener.service_listener.ServiceListener(services=None)``Bases: object``r_pool``r_conn``registration_poll`

```

listener_poll
processor_poll
processing_limit
processing_queue_key
processing_queue_key_timeout
processing_lock_key
processing_lock_key_timeout
preprocessing_lock_key
preprocessing_lock_key_timeout
preprocessing_worker_lock_key
preprocessing_worker_lock_key_timeout
registration_channel
classmethod publish_response(cls, channel, data)

```

Return type `bool`

```

start_workers(self)
stop_workers(self)
pre_process_message(self, message_channel, message_data)

```

Return type `bool | dict`

```

static process_message(job_details)
_start_registration_task(self)
_start_channel_task(self)
_start_listener_task(self)
_start_queue_processor_task(self)

```

`pine.pipelines.shared`

Submodules

`pine.pipelines.shared.config`

Module Contents

Classes

<code>BaseConfig(root_dir=None)</code>	Initialize self. See help(type(self)) for accurate signature.
<code>TestConfig(root_dir=None)</code>	Initialize self. See help(type(self)) for accurate signature.
<code>ConfigBuilder()</code>	Initialize self. See help(type(self)) for accurate signature.

`pine.pipelines.shared.config.LOGGER`

```
class pine.pipelines.shared.config.BaseConfig (root_dir=None)
    Bases: object

    Initialize self. See help(type(self)) for accurate signature.

    ROOT_DIR
    BASE_DIR
    BASE_CFG_FILE = config.yaml
    BASE_ENV_PREFIX = AL_
    DEBUG = True
    TESTING = False
    LOGGER_NAME
    LOGGER_DIR = logs
    LOGGER_FILE = debug.log
    LOGGER_LEVEL
    PIPELINE = opennlp
    EVE_HOST = localhost
    EVE_PORT = 5001
    REDIS_HOST = localhost
    REDIS_PORT = 6379
    REDIS_USR
    REDIS_PWD
    REDIS_DBNUM = 0
    REDIS_PREFIX = AL:
    REDIS_EXPIRE = 3600
    SCHEDULER_REGISTRATION_TIMEOUT
    SCHEDULER_HANDLER_TIMEOUT
    SCHEDULER_QUEUE_TIMEOUT
    SERVICE_REGISTRATION_CHANNEL = registration
    SERVICE_REGISTRATION_FREQUENCY = 60
    SERVICE_LISTENING_FREQUENCY = 1
    SERVICE_HANDLER_TIMEOUT = 60
    SERVICE_LIST
    MODELS_DIR

    classmethod _get_config_var_paths (cls, root_dict=None)
    classmethod _process_paths (cls, alt_path=None)
    classmethod _process_file_cfg (cls)
    classmethod _process_env_vars (cls)
```



```

classmethod as_dict (cls)
    Return type dict

classmethod as_attr_dict (cls)
    Return type munch.Munch | dict

static _try_cast (value, _type, _default=None)

static _str2bool (_str, _default=None)

class pine.pipelines.shared.config.TestConfig (root_dir=None)
    Bases: pine.pipelines.shared.config.BaseConfig

    Initialize self. See help(type(self)) for accurate signature.

class pine.pipelines.shared.config.ConfigBuilder
    Bases: object

    Initialize self. See help(type(self)) for accurate signature.

    __env_cfg_variable = BUILDER_CFG_PROFILE

    __current_config_instance

    __current_config_instance_name

    __current_config_instance_print = False

    __arg_parser

    static __get_configs ()
        :rtype list[Callable[... BaseConfig]]

    static get_config_names ()
        Return type list[str]

    classmethod get_arg_parser (cls)
        Return type ArgumentParser

    classmethod init_config (cls, config_name=None, config_base=None, enable_terminal=True,
        as_attr_dict=True)

    classmethod get_config (cls, config_name=None, config_base=None, enable_terminal=True,
        as_attr_dict=True)
        Return type BaseConfig

    classmethod set_config (cls, config_name=None, config_base=None, enable_terminal=True,
        as_attr_dict=True)

    classmethod __parse_terminal_config (cls)
        Return type argparse.Namespace

```

`pine.pipelines.shared.transform`

Module Contents

Functions

<code>transform_module_by_config(module_ref, config_ref, config_prefix=None)</code>	Transforms a given module's properties based on ConfigBuilder Values.
---	---

`pine.pipelines.shared.transform.transform_module_by_config` (*module_ref*, *config_ref*, *config_prefix=None*)

Transforms a given module's properties based on ConfigBuilder Values. The prefix can be used to avoid blindly changing values and target a subset of matching values in `config_ref`. :type module_ref: ModuleType :type config_ref: dict :type config_prefix: str

Submodules

`pine.pipelines.EveClient`

Module Contents

Classes

<code>EveClient</code> (<i>entry_point='{ }:{ }'.format(config.EVE_HOST,Initialize self. See help(type(self)) for accurate signature. config.EVE_PORT)</i>)

`pine.pipelines.EveClient.logger`

`pine.pipelines.EveClient.config`

class `pine.pipelines.EveClient.EveClient` (*entry_point='{ }:{ }'.format(config.EVE_HOST, config.EVE_PORT)*)

Bases: `object`

Initialize self. See help(type(self)) for accurate signature.

eve_headers

add (*self, resource, add_object*)

get_obj (*self, resource, id*)

get_all_items (*self, resource*)

get_all_ids (*self, resource*)

get_items (*self, resource*)

get_documents (*self, collection_id*)

get_docs_with_annotations (*self, collection_id, doc_map*)

update (*self, resource, id, etag, update_obj*)

`pine.pipelines.NERWrapper`

Module Contents

Classes

<code>NERWrapper</code> (<i>classifier_type</i> , <i>model_dir</i> ='models', <i>entry_point</i> ='localhost:5000')	Initialize self. See help(type(self)) for accurate signature.
--	---

```
class pine.pipelines.NERWrapper.NERWrapper (classifier_type, model_dir='models', entry_point='localhost:5000')
    Initialize self. See help(type(self)) for accurate signature.

    eve_headers
    load_classifier (self, classifier_id)
    predict (self, classifier_id, documents, document_ids)
    update_model (self, classifier_id)
    update (self, resource, id, etag, update_obj)
    get_obj (self, resource, id)
    get_all_items (self, resource)
    get_all_ids (self, resource)
    get_items (self, resource)
```

`pine.pipelines.NERWrapper.parser`

`pine.pipelines.NER_API`

Module Contents

Classes

<code>ner_api</code> ()	Initialize self. See help(type(self)) for accurate signature.
-------------------------	---

`pine.pipelines.NER_API.logger`

`pine.pipelines.NER_API.config`

```
class pine.pipelines.NER_API.ner_api
    Bases: object
    Initialize self. See help(type(self)) for accurate signature.

    perform_fold (self, model, train_data, test_data, pipeline_parameters)
    perform_five_fold (self, model, documents, annotations, doc_ids, pipeline_parameters)
    get_document_ranking (self, model, doc_map, doc_ids)
    get_classifier_pipeline_metrics_objs (self, classifier_id)
```

```
train_model (self, custom_filename, classifier_id, pipeline_name)
predict (self, classifier_id, pipeline_name, documents, document_ids)
```

`pine.pipelines.RankingFunctions`

Module Contents

Functions

<code>rank(results, metric)</code>	if metric == 'lc': return least_confidence(results)
<code>least_confidence(results)</code>	
<code>least_confidence_squared(results)</code>	
<code>least_confidence_squared_by_entity(results)</code>	
<code>largest_margin(results)</code>	
<code>entropy_rank(results, N=None)</code>	
<code>random_rank(results)</code>	
<code>most_of_least_popular(results)</code>	

```
pine.pipelines.RankingFunctions.rank (results, metric)
    if metric == 'lc': return least_confidence(results) if metric == 'ma': return largest_margin(results) if metric ==
    'en': return entropy_rank(results) if metric == 'lcs': return least_confidence_squared(results) if metric == 'lce':
    return least_confidence_squared_by_entity(results) if metric == 'ra': return random_rank(results) if metric ==
    'mlp': return most_of_least_popular(results) return -1
```

```
#Dictionary method is inefficient as it runs every method before returning one
```

```
pine.pipelines.RankingFunctions.least_confidence (results)
pine.pipelines.RankingFunctions.least_confidence_squared (results)
pine.pipelines.RankingFunctions.least_confidence_squared_by_entity (results)
pine.pipelines.RankingFunctions.largest_margin (results)
pine.pipelines.RankingFunctions.entropy_rank (results, N=None)
pine.pipelines.RankingFunctions.random_rank (results)
pine.pipelines.RankingFunctions.most_of_least_popular (results)
```

`pine.pipelines.corenlp_NER_pipeline`

Module Contents

Classes

<code>corenlp_NER(java_dir=None, ner_path=None, load_model=None, tmp_dir=None)</code>	Initialize self. See help(type(self)) for accurate signature.
---	---

```
pine.pipelines.corenlp_NER_pipeline.config
pine.pipelines.corenlp_NER_pipeline.logger
```

```
class pine.pipelines.corenlp_NER_pipeline.corenlp_NER (java_dir=None,  
                                                    ner_path=None,  
                                                    load_model=None,  
                                                    tmp_dir=None)
```

Bases: *pine.pipelines.pipeline.Pipeline*

Initialize self. See help(type(self)) for accurate signature.

```
__jar =  
__jdk_dir =  
__train_file =  
__test_file =  
__model =  
__temp_dir  
__crf  
__props  
__id  
fit (self, X, y, params=None)  
predict (self, X, Xid)  
predict_proba (self, X, Xid, get_all_labels=False, include_other=False)  
next_example (self, X, Xid)  
get_id (self)  
format_data (self, X, y)  
save_model (self, model_name)  
load_model (self, model_name)  
tokenize (self, input_text)  
evaluate (self, X, y, Xid, verbose=False)  
evaluate_orig (self, X, y, Xid)
```

pine.pipelines.opennlp_NER_pipeline

Module Contents

Classes

<i>opennlp_NER</i> (<i>java_dir=None,</i>	<i>ner_path=None,</i>	Initialize self. See help(type(self)) for accurate signature.
<i>tmp_dir=None</i>)		

pine.pipelines.opennlp_NER_pipeline.**config**

pine.pipelines.opennlp_NER_pipeline.**logger**

```
class pine.pipelines.opennlp_NER_pipeline.opennlp_NER (java_dir=None,  
ner_path=None,  
tmp_dir=None)
```

Bases: *pine.pipelines.pipeline.Pipeline*

Initialize self. See help(type(self)) for accurate signature.

```
__jar =  
__jdk_dir =  
__temp_dir  
__train_file =  
__test_file =  
__model  
__nameFinder  
__id  
fit (self, X, y, params)  
predict (self, X, Xid)  
predict_proba (self, X, Xid, get_all_labels=False, include_other=False)  
next_example (self, X, Xid)  
save_model (self, model_name)  
load_model (self, model_name)  
find_all_init (self)  
find_all (self, tokens)  
get_id (self)  
format_data (self, X, y)  
convert_ann_collection_to_per_token (self, annotations, tokens)  
evaluate (self, X, y, Xid)  
evaluate_orig (self, X, y, Xid)
```

pine.pipelines.pipeline

Module Contents

Classes

<i>Pipeline</i> ()	Initialize self. See help(type(self)) for accurate signature.
--------------------	---

```
class pine.pipelines.pipeline.Pipeline
```

Bases: *object*

Initialize self. See help(type(self)) for accurate signature.

```
abstract fit (self, X, y)
```

```

abstract predict (self, X, Xid)
abstract predict_proba (self, X, Xid)
abstract next_example (self, X, Xid)
abstract save_model (self, model_name)
abstract load_model (self, model_name)

```

`pine.pipelines.pmap_ner`

Module Contents

Classes

<code>NER(lib=None, **kwargs)</code>	Initialize self. See help(type(self)) for accurate signature.
--------------------------------------	---

`pine.pipelines.pmap_ner.logger`

```

class pine.pipelines.pmap_ner.NER (lib=None, **kwargs)
    Bases: pine.pipelines.pipeline.Pipeline
    Initialize self. See help(type(self)) for accurate signature.
    __lib =
    pipeline
    __SUPPORTED_PIPELINES = ['spacy', 'corenlp', 'opennlp']
    pipe_init (self, x, **kwargs)
    fit (self, X, y, kwargs)
    predict (self, X, Xid)
    predict_proba (self, X, Xid, **kwargs)
    evaluate (self, X, y, Xid, **kwargs)
    next_example (self, X, Xid)
    save_model (self, model_path)
    load_model (self, model_name)

```

`pine.pipelines.run_service`

`pine.pipelines.spacy_NER_pipeline`

For more details, see the documentation: * Training: <https://spacy.io/usage/training> * NER: <https://spacy.io/usage/linguistic-features#named-entities>

Compatible with: spaCy v2.0.0+

Module Contents

Classes

<code>spacy_NER(model_path=None)</code>	Initialize self. See help(type(self)) for accurate signature.
---	---

class pine.pipelines.spacy_NER_pipeline.**spacy_NER** (*model_path=None*)

Bases: `pine.pipelines.pipeline.Pipeline`

Initialize self. See help(type(self)) for accurate signature.

`__nlp = []`

`__ner = []`

`__optimizer = []`

`fit (self, X, y, params=None)`

`evaluate (self, X, y, Xid)`

`predict (self, X, Xid)`

`predict_proba (self, X, Xid)`

`next_example (self, X, Xid)`

`format_data (self, X, y)`

`add_label (self, entity)`

`save_model (self, model_path)`

`load_model (self, model_path)`

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

- pine, 1
- pine.backend, 1
 - pine.backend.admin, 1
 - pine.backend.admin.bp, 1
 - pine.backend.annotations, 2
 - pine.backend.annotations.bp, 2
 - pine.backend.app, 28
 - pine.backend.auth, 4
 - pine.backend.auth.bp, 4
 - pine.backend.auth.eve, 5
 - pine.backend.auth.oauth, 6
 - pine.backend.auth.password, 7
 - pine.backend.auth.vegas, 7
 - pine.backend.collections, 8
 - pine.backend.collections.bp, 8
 - pine.backend.config, 28
 - pine.backend.cors, 28
 - pine.backend.data, 12
 - pine.backend.data.bp, 12
 - pine.backend.data.service, 12
 - pine.backend.data.users, 14
 - pine.backend.documents, 15
 - pine.backend.documents.bp, 15
 - pine.backend.job_manager, 16
 - pine.backend.job_manager.service, 16
 - pine.backend.log, 29
 - pine.backend.models, 30
 - pine.backend.pineiaa, 18
 - pine.backend.pineiaa.bp, 23
 - pine.backend.pineiaa.bratiaa, 18
 - pine.backend.pineiaa.bratiaa.agree, 18
 - pine.backend.pineiaa.bratiaa.agree_cli, 20
 - pine.backend.pineiaa.bratiaa.evaluation, 20
 - pine.backend.pineiaa.bratiaa.iaa_service, 21
 - pine.backend.pineiaa.bratiaa.utils, 21
 - pine.backend.pipelines, 24
 - pine.backend.pipelines.bp, 24
 - pine.backend.shared, 25
 - pine.backend.shared.config, 25
 - pine.backend.shared.transform, 27
 - pine.client, 31
 - pine.client.client, 31
 - pine.client.exceptions, 43
 - pine.client.log, 44
 - pine.client.models, 44
 - pine.client.password, 55
 - pine.pipelines, 65
 - pine.pipelines.app, 65
 - pine.pipelines.app.listener, 65
 - pine.pipelines.app.listener.main, 65
 - pine.pipelines.app.listener.service_listener, 66
 - pine.pipelines.corenlp_NER_pipeline, 72
 - pine.pipelines.EveClient, 70
 - pine.pipelines.NER_API, 71
 - pine.pipelines.NERWrapper, 71
 - pine.pipelines.opennlp_NER_pipeline, 73
 - pine.pipelines.pipeline, 74
 - pine.pipelines.pmap_ner, 75
 - pine.pipelines.RankingFunctions, 72
 - pine.pipelines.run_service, 75
 - pine.pipelines.shared, 67
 - pine.pipelines.shared.config, 67
 - pine.pipelines.shared.transform, 70
 - pine.pipelines.spacy_NER_pipeline, 75

Symbols

__SUPPORTED_PIPELINES
 (*pine.pipelines.pmap_ner.NER* attribute), 75
 __arg_parser (*pine.backend.shared.config.ConfigBuilder* attribute), 27
 __arg_parser (*pine.pipelines.shared.config.ConfigBuilder* attribute), 69
 __crf (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 73
 __current_config_instance
 (*pine.backend.shared.config.ConfigBuilder* attribute), 26
 __current_config_instance
 (*pine.pipelines.shared.config.ConfigBuilder* attribute), 69
 __current_config_instance_name
 (*pine.backend.shared.config.ConfigBuilder* attribute), 26
 __current_config_instance_name
 (*pine.pipelines.shared.config.ConfigBuilder* attribute), 69
 __current_config_instance_print
 (*pine.backend.shared.config.ConfigBuilder* attribute), 27
 __current_config_instance_print
 (*pine.pipelines.shared.config.ConfigBuilder* attribute), 69
 __env_cfg_variable
 (*pine.backend.shared.config.ConfigBuilder* attribute), 26
 __env_cfg_variable
 (*pine.pipelines.shared.config.ConfigBuilder* attribute), 69
 __get_configs() (*pine.backend.shared.config.ConfigBuilder* static method), 27
 __get_configs() (*pine.pipelines.shared.config.ConfigBuilder* static method), 69
 __id (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 73
 __id (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 74
 __jar (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 73
 __jar (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 74
 __jdk_dir (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 73
 __jdk_dir (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 74
 __lib (*pine.pipelines.pmap_ner.NER* attribute), 75
 __metaclass__ (*pine.backend.auth.bp.AuthModule* attribute), 5
 __metaclass__ (*pine.client.client.BaseClient* attribute), 32
 __model (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 73
 __model (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 74
 __nameFinder (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 74
 __ner (*pine.pipelines.spacy_NER_pipeline.spacy_NER* attribute), 76
 __nlp (*pine.pipelines.spacy_NER_pipeline.spacy_NER* attribute), 76
 __optimizer (*pine.pipelines.spacy_NER_pipeline.spacy_NER* attribute), 76
 __parse_terminal_config()
 (*pine.backend.shared.config.ConfigBuilder* class method), 27
 __parse_terminal_config()
 (*pine.pipelines.shared.config.ConfigBuilder* class method), 69
 __props (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 73
 __slots__ (*pine.backend.pineiaa.bratiaa.Document* attribute), 23
 __slots__ (*pine.backend.pineiaa.bratiaa.agree.Document* attribute), 19
 __temp_dir (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* attribute), 73
 __temp_dir (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* attribute), 74
 __test_file (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER*

[attribute](#)), 73
[__test_file](#) ([pine.pipelines.opennlp_NER_pipeline.opennlp_NER_pipeline](#) [attribute](#)), 74
[__train_file](#) ([pine.pipelines.corenlp_NER_pipeline.corenlp_NER_pipeline](#) [attribute](#)), 73
[__train_file](#) ([pine.pipelines.opennlp_NER_pipeline.opennlp_NER_pipeline](#) [attribute](#)), 74
[__version__](#) (in module [pine.backend](#)), 31
[_add_or_replace_resource](#) ([pine.client.client.EveClient](#) method), 35
[_add_or_update_annotation](#) (in module [pine.backend.annotations.bp](#)), 4
[_add_resources](#) ([pine.client.client.EveClient](#) method), 36
[_channel_watchdog](#) ([pine.backend.job_manager.service.ServiceManager](#) method), 18
[_check_collection_and_get_image_dir](#) (in module [pine.backend.collections.bp](#)), 10
[_check_documents](#) (in module [pine.backend.documents.bp](#)), 16
[_check_field_bool](#) (in module [pine.client.models](#)), 48
[_check_field_dict](#) (in module [pine.client.models](#)), 47
[_check_field_float](#) (in module [pine.client.models](#)), 46
[_check_field_int](#) (in module [pine.client.models](#)), 46
[_check_field_required_bool](#) (in module [pine.client.models](#)), 45
[_check_field_required_dict](#) (in module [pine.client.models](#)), 47
[_check_field_required_float](#) (in module [pine.client.models](#)), 46
[_check_field_required_int](#) (in module [pine.client.models](#)), 46
[_check_field_required_string](#) (in module [pine.client.models](#)), 47
[_check_field_required_string_list](#) (in module [pine.client.models](#)), 47
[_check_field_string](#) (in module [pine.client.models](#)), 46
[_check_field_string_list](#) (in module [pine.client.models](#)), 47
[_check_login](#) ([pine.client.PineClient](#) method), 57
[_check_login](#) ([pine.client.client.PineClient](#) method), 38
[_collection_user_can](#) (in module [pine.backend.collections.bp](#)), 9
[_collection_user_can_projection](#) (in module [pine.backend.collections.bp](#)), 9
[_compute_tp_fp_fn](#) ([pine.backend.pineiaa.bratiaa.F1Agreement](#) method), 22
[_compute_tp_fp_fn](#) ([pine.backend.pineiaa.bratiaa.F1Agreement](#) method), 19
[_document_user_can_projection](#) (in module [pine.backend.documents.bp](#)), 15
[_get_classifier](#) (in module [pine.backend.pipelines.bp](#)), 25
[_get_classifier_metrics](#) (in module [pine.backend.pipelines.bp](#)), 24
[_get_collection_classifier](#) (in module [pine.backend.pipelines.bp](#)), 24
[_get_config_var_paths](#) ([pine.backend.shared.config.BaseConfig](#) class method), 26
[_get_config_var_paths](#) ([pine.pipelines.shared.config.BaseConfig](#) class method), 68
[_get_next_instance](#) (in module [pine.backend.pipelines.bp](#)), 25
[_increment_counts](#) ([pine.backend.pineiaa.bratiaa.F1Agreement](#) method), 22
[_increment_counts](#) ([pine.backend.pineiaa.bratiaa.agree.F1Agreement](#) method), 19
[_make_annotations](#) (in module [pine.backend.annotations.bp](#)), 3
[_mean_sd](#) ([pine.backend.pineiaa.bratiaa.F1Agreement](#) static method), 23
[_mean_sd](#) ([pine.backend.pineiaa.bratiaa.agree.F1Agreement](#) static method), 19
[_pairs_involving](#) ([pine.backend.pineiaa.bratiaa.F1Agreement](#) method), 23
[_pairs_involving](#) ([pine.backend.pineiaa.bratiaa.agree.F1Agreement](#) method), 19
[_path_split](#) (in module [pine.backend.collections.bp](#)), 11
[_process_env_vars](#) ([pine.backend.shared.config.BaseConfig](#) class method), 26
[_process_env_vars](#) ([pine.pipelines.shared.config.BaseConfig](#) class method), 68
[_process_file_cfg](#) ([pine.backend.shared.config.BaseConfig](#) class method), 26
[_process_file_cfg](#) ([pine.pipelines.shared.config.BaseConfig](#) class method), 68
[_process_paths](#) ([pine.backend.shared.config.BaseConfig](#) class method), 26

[_process_paths\(\)](#) (*pine.pipelines.shared.config.BaseConfig* static method), 69
[_processing_listener\(\)](#) (*pine.backend.job_manager.service.ServiceManager* class method), 68
[_processing_listener_handler_wrapper\(\)](#) (*pine.backend.job_manager.service.ServiceManager* class method), 18
[_read_token_annotations\(\)](#) (*pine.backend.pineiaa.bratiaa.evaluation*), 20
[_registration_listener\(\)](#) (*pine.backend.job_manager.service.ServiceManager* class method), 17
[_req\(\)](#) (*pine.client.client.BaseClient* method), 33
[_safe_path\(\)](#) (*pine.backend.collections.bp*), 11
[_standardize_path\(\)](#) (*pine.backend.data.service*), 13
[_standardize_path\(\)](#) (*pine.client.client*), 32
[_start_channel_task\(\)](#) (*pine.pipelines.app.listener.service_listener.ServiceListener* method), 67
[_start_channel_watchdog\(\)](#) (*pine.backend.job_manager.service.ServiceManager* class method), 17
[_start_listener_task\(\)](#) (*pine.pipelines.app.listener.service_listener.ServiceListener* method), 67
[_start_processing_listeners\(\)](#) (*pine.backend.job_manager.service.ServiceManager* class method), 17
[_start_queue_processor_task\(\)](#) (*pine.pipelines.app.listener.service_listener.ServiceListener* method), 67
[_start_registration_listener\(\)](#) (*pine.backend.job_manager.service.ServiceManager* class method), 17
[_start_registration_task\(\)](#) (*pine.pipelines.app.listener.service_listener.ServiceListener* method), 67
[_stop_channel_watchdog\(\)](#) (*pine.backend.job_manager.service.ServiceManager* class method), 17
[_str2bool\(\)](#) (*pine.backend.shared.config.BaseConfig* static method), 26
[_str2bool\(\)](#) (*pine.pipelines.shared.config.BaseConfig* static method), 69
[_thread_killer\(\)](#) (*pine.backend.job_manager.service.ServiceManager* static method), 18
[_try_cast\(\)](#) (*pine.backend.shared.config.BaseConfig* static method), 26
[_try_cast\(\)](#) (*pine.pipelines.shared.config.BaseConfig* static method), 69
[_upload_collection_image_file\(\)](#) (*pine.backend.collections.bp*), 11
[_upload_documents\(\)](#) (*pine.backend.collections.bp*), 10
[about\(\)](#) (*pine.client.client.EveClient* method), 35
[about\(\)](#) (*pine.client.client.PineClient* method), 37
[about\(\)](#) (*pine.client.PineClient* method), 56
[access\(\)](#) (*pine.backend.log*), 30
[access_flask_add_collection\(\)](#) (*pine.backend.log*), 30
[access_flask_add_document\(\)](#) (*pine.backend.log*), 30
[access_flask_add_documents\(\)](#) (*pine.backend.log*), 30
[access_flask_annotate_document\(\)](#) (*pine.backend.log*), 30
[access_flask_annotate_documents\(\)](#) (*pine.backend.log*), 30
[access_flask_login\(\)](#) (*pine.backend.log*), 30
[access_flask_logout\(\)](#) (*pine.backend.log*), 30
[access_flask_view_document\(\)](#) (*pine.backend.log*), 30
[ACCESS_LOGGER](#) (*pine.backend.log*), 29
[ACCESS_LOGGER_NAME](#) (*pine.backend.log*), 29
[Action](#) (*pine.backend.log*), 29
[add\(\)](#) (*pine.backend.data.service.PerformanceHistory* method), 13
[add\(\)](#) (*pine.pipelines.EveClient.EveClient* method), 70
[add_admin_command\(\)](#) (*pine.backend.data.users*), 15
[add_annotator_to_collection\(\)](#) (*pine.backend.collections.bp*), 10
[ADD_DOCUMENT](#) (*pine.backend.log.Action* attribute), 29
[add_document\(\)](#) (*pine.backend.documents.bp*), 16
[add_document\(\)](#) (*pine.client.client.PineClient* method), 39
[add_document\(\)](#) (*pine.client.PineClient* method), 58
[ADD_DOCUMENTS](#) (*pine.backend.log.Action* attribute), 29
[add_documents\(\)](#) (*pine.client.client.PineClient* method), 40
[add_documents\(\)](#) (*pine.client.PineClient* method), 58
[add_label\(\)](#) (*pine.pipelines.spacy_NER_pipeline.spacy_NER* method), 76
[add_label_to_collection\(\)](#) (*pine.backend.collections.bp*), 10

`add_pipelines()` (*pine.client.client.EveClient method*), 36
`add_user()` (*in module pine.backend.admin.bp*), 2
`add_users()` (*pine.client.client.EveClient method*), 36
`add_viewer_to_collection()` (*in module pine.backend.collections.bp*), 10
`admin_required()` (*in module pine.backend.auth*), 7
`admin_required()` (*in module pine.backend.auth.bp*), 4
`advance_to_next_document_by_classifier()` (*in module pine.backend.pipelines.bp*), 25
`allow_overlapping_ner_annotations()` (*pine.client.CollectionBuilder method*), 64
`allow_overlapping_ner_annotations()` (*pine.client.models.CollectionBuilder method*), 53
`AnnFile` (*in module pine.backend.pineiaa.bratiaa*), 22
`AnnFile` (*in module pine.backend.pineiaa.bratiaa.agree*), 18
`annotate_collection_documents()` (*pine.client.client.PineClient method*), 41
`annotate_collection_documents()` (*pine.client.PineClient method*), 60
`ANNOTATE_DOCUMENT` (*pine.backend.log.Action attribute*), 30
`annotate_document()` (*pine.client.client.PineClient method*), 40
`annotate_document()` (*pine.client.PineClient method*), 59
`ANNOTATE_DOCUMENTS` (*pine.backend.log.Action attribute*), 30
`Annotation` (*in module pine.backend.pineiaa.bratiaa*), 23
`Annotation` (*in module pine.backend.pineiaa.bratiaa.agree*), 18
`Annotation` (*in module pine.backend.pineiaa.bratiaa.evaluation*), 20
`annotator()` (*pine.client.CollectionBuilder method*), 63
`annotator()` (*pine.client.models.CollectionBuilder method*), 52
`annotators()` (*pine.backend.pineiaa.bratiaa.agree.FIAgreement property*), 19
`annotators()` (*pine.backend.pineiaa.bratiaa.FIAgreement property*), 22
`archive_collection()` (*in module pine.backend.collections.bp*), 10
`archive_or_unarchive_collection()` (*in module pine.backend.collections.bp*), 10
`as_attr_dict()` (*pine.backend.shared.config.BaseConfig class method*), 26
`as_attr_dict()` (*pine.pipelines.shared.config.BaseConfig class method*), 69
`as_dict()` (*pine.backend.shared.config.BaseConfig class method*), 26
`as_dict()` (*pine.pipelines.shared.config.BaseConfig class method*), 68
`AUTH_MODULE` (*in module pine.backend.config*), 28
`AuthModule` (*class in pine.backend.auth.bp*), 5
`authorize()` (*pine.backend.auth.oauth.OAuthModule method*), 6
`AuthUser` (*class in pine.backend.models*), 31

B

`backoff_hdlr()` (*in module pine.pipelines.app.listener.main*), 66
`BASE_CFG_FILE` (*pine.backend.shared.config.BaseConfig attribute*), 25
`BASE_CFG_FILE` (*pine.pipelines.shared.config.BaseConfig attribute*), 68
`BASE_DIR` (*pine.backend.shared.config.BaseConfig attribute*), 25
`BASE_DIR` (*pine.pipelines.shared.config.BaseConfig attribute*), 68
`BASE_ENV_PREFIX` (*pine.backend.shared.config.BaseConfig attribute*), 25
`BASE_ENV_PREFIX` (*pine.pipelines.shared.config.BaseConfig attribute*), 68
`base_uri` (*pine.client.client.BaseClient attribute*), 32
`BaseClient` (*class in pine.client.client*), 32
`BaseConfig` (*class in pine.backend.shared.config*), 25
`BaseConfig` (*class in pine.pipelines.shared.config*), 67
`bp` (*in module pine.backend.admin.bp*), 2
`bp` (*in module pine.backend.annotations.bp*), 3
`bp` (*in module pine.backend.auth.bp*), 4
`bp` (*in module pine.backend.collections.bp*), 9
`bp` (*in module pine.backend.documents.bp*), 15
`bp` (*in module pine.backend.pineiaa.bp*), 24
`bp` (*in module pine.backend.pipelines.bp*), 24

C

`can_annotate_document()` (*in module pine.backend.documents.bp*), 16
`can_manage_users()` (*pine.backend.auth.bp.AuthModule method*), 5
`can_manage_users()` (*pine.backend.auth.eve.EveModule method*), 5
`can_manage_users()` (*pine.backend.auth.oauth.OAuthModule method*), 6
`can_modify_metadata()` (*in module pine.backend.documents.bp*), 16
`channel_worker_name` (*pine.backend.job_manager.service.ServiceManager attribute*), 17
`check_document()` (*in module pine.backend.annotations.bp*), 3

check_document_by_id() (in module *pine.backend.annotations.bp*), 3
 check_overlapping_annotations() (in module *pine.backend.annotations.bp*), 3
 check_password() (in module *pine.backend.auth.password*), 7
 check_password() (in module *pine.client.password*), 55
 classifier() (*pine.client.CollectionBuilder* method), 64
 classifier() (*pine.client.models.CollectionBuilder* method), 53
 classifier_dict (in module *pine.backend.pipelines.bp*), 24
 classifier_pipelines (in module *pine.backend.pipelines.bp*), 24
 collection() (*pine.client.CollectionBuilder* property), 63
 collection() (*pine.client.models.CollectionBuilder* property), 52
 collection_builder() (*pine.client.client.PineClient* method), 39
 collection_builder() (*pine.client.PineClient* method), 58
 CollectionBuilder (class in *pine.client*), 61
 CollectionBuilder (class in *pine.client.models*), 51
 compute_f1() (in module *pine.backend.pineiaa.bratkaa.agree*), 19
 compute_f1_agreement() (in module *pine.backend.pineiaa.bratkaa*), 22
 compute_f1_agreement() (in module *pine.backend.pineiaa.bratkaa.agree*), 19
 compute_mapping() (*pine.backend.pineiaa.bratkaa.utils.TokenOverlap* static method), 22
 compute_total_f1_matrix() (*pine.backend.pineiaa.bratkaa.agree.F1Agreement* method), 19
 compute_total_f1_matrix() (*pine.backend.pineiaa.bratkaa.F1Agreement* method), 23
 config (in module *pine.backend.job_manager.service*), 16
 config (in module *pine.pipelines.app.listener.service_listener*), 66
 config (in module *pine.pipelines.corenlp_NER_pipeline*), 72
 config (in module *pine.pipelines.EveClient*), 70
 config (in module *pine.pipelines.NER_API*), 71
 config (in module *pine.pipelines.opennlp_NER_pipeline*), 73
 CONFIG_ALLOW_OVERLAPPING_NER_ANNOTATIONS (in module *pine.backend.annotations.bp*), 3
 CONFIG_AUTH_MODULE_KEY (in module *pine.backend.auth.bp*), 4
 CONFIG_FILE_ENV (in module *pine.backend.log*), 29
 CONFIG_FILE_ENV (in module *pine.client.log*), 44
 ConfigBuilder (class in *pine.backend.shared.config*), 26
 ConfigBuilder (class in *pine.pipelines.shared.config*), 69
 configuration() (*pine.client.CollectionBuilder* method), 64
 configuration() (*pine.client.models.CollectionBuilder* method), 53
 convert_ann_collection_to_per_token() (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* method), 74
 convert_response() (in module *pine.backend.data.service*), 14
 corenlp_NER (class in *pine.pipelines.corenlp_NER_pipeline*), 72
 counter2list() (in module *pine.backend.pineiaa.bratkaa.evaluation*), 20
 create_app() (in module *pine.backend*), 31
 create_app() (in module *pine.backend.app*), 28
 CREATE_COLLECTION (*pine.backend.log.Action* attribute), 29
 create_collection() (in module *pine.backend.collections.bp*), 10
 create_collection() (*pine.client.client.PineClient* method), 39
 create_collection() (*pine.client.PineClient* method), 58
 create_iaa_report_by_collection_id() (in module *pine.backend.pineiaa.bp*), 24
 creator_id() (*pine.client.CollectionBuilder* method), 63
 creator_id() (*pine.client.models.CollectionBuilder* method), 52

D

DATASETS_LOCAL_DIR (*pine.backend.shared.config.BaseConfig* attribute), 26
 DEBUG (in module *pine.backend.config*), 28
 DEBUG (*pine.backend.shared.config.BaseConfig* attribute), 25
 DEBUG (*pine.pipelines.shared.config.BaseConfig* attribute), 68
 DEFAULT_DBNAME (*pine.client.client.EveClient* attribute), 35
 delete() (in module *pine.backend.data.service*), 14
 delete_user() (in module *pine.backend.admin.bp*), 2
 description() (*pine.client.CollectionBuilder* method), 64

[description\(\)](#) (*pine.client.models.CollectionBuilder method*), 53
[display_name\(\)](#) (*pine.backend.auth.eve.EveUser property*), 5
[display_name\(\)](#) (*pine.backend.auth.oauth.OAuthUser property*), 6
[display_name\(\)](#) (*pine.backend.models.AuthUser property*), 31
[Document](#) (*class in pine.backend.pineiaa.brataia*), 23
[Document](#) (*class in pine.backend.pineiaa.brataia.agree*), 18
[document_csv_file\(\)](#) (*pine.client.CollectionBuilder method*), 65
[document_csv_file\(\)](#) (*pine.client.models.CollectionBuilder method*), 54
[DOCUMENT_IMAGE_DIR](#) (*in module pine.backend.config*), 28
[documents\(\)](#) (*pine.backend.pineiaa.brataia.agree.FIAgreement property*), 19
[documents\(\)](#) (*pine.backend.pineiaa.brataia.FIAgreement property*), 22
[DOCUMENTS_PER_TRANSACTION](#) (*in module pine.backend.collections.bp*), 9
[download_collection\(\)](#) (*in module pine.backend.collections.bp*), 10
[download_collection_data\(\)](#) (*pine.client.client.PineClient method*), 41
[download_collection_data\(\)](#) (*pine.client.PineClient method*), 60
[draw_heatmap\(\)](#) (*pine.backend.pineiaa.brataia.agree.FIAgreement method*), 19
[draw_heatmap\(\)](#) (*pine.backend.pineiaa.brataia.FIAgreement method*), 23
E
[ENCODING](#) (*in module pine.backend.pineiaa.brataia.utils*), 21
[entropy_rank\(\)](#) (*in module pine.pipelines.RankingFunctions*), 72
[evaluate\(\)](#) (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER method*), 73
[evaluate\(\)](#) (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER method*), 74
[evaluate\(\)](#) (*pine.pipelines.pmap_ner.NER method*), 75
[evaluate\(\)](#) (*pine.pipelines.spacy_NER_pipeline.spacy_NER method*), 76
[evaluate_orig\(\)](#) (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER method*), 73
[evaluate_orig\(\)](#) (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER method*), 74
[eve](#) (*pine.client.client.LocalPineClient attribute*), 42
[eve](#) (*pine.client.LocalPineClient attribute*), 61
[EVE_HEADERS](#) (*in module pine.backend.pineiaa.brataia.iaa_service*), 21
[eve_headers](#) (*pine.pipelines.EveClient.EveClient attribute*), 70
[eve_headers](#) (*pine.pipelines.NERWrapper.NERWrapper attribute*), 71
[EVE_HOST](#) (*pine.backend.shared.config.BaseConfig attribute*), 25
[EVE_HOST](#) (*pine.pipelines.shared.config.BaseConfig attribute*), 68
[EVE_PORT](#) (*pine.backend.shared.config.BaseConfig attribute*), 26
[EVE_PORT](#) (*pine.pipelines.shared.config.BaseConfig attribute*), 68
[EVE_SERVER](#) (*in module pine.backend.config*), 28
[EveClient](#) (*class in pine.client.client*), 34
[EveClient](#) (*class in pine.pipelines.EveClient*), 70
[EveUser](#) (*class in pine.backend.auth.eve*), 5
[exact_match_instance_evaluation\(\)](#) (*in module pine.backend.pineiaa.brataia*), 23
[exact_match_instance_evaluation\(\)](#) (*in module pine.backend.pineiaa.brataia.evaluation*), 20
[exact_match_token_evaluation\(\)](#) (*in module pine.backend.pineiaa.brataia*), 23
[exact_match_token_evaluation\(\)](#) (*in module pine.backend.pineiaa.brataia.evaluation*), 20
[FIAgreement](#) (*class in pine.backend.pineiaa.brataia*), 22
[FIAgreement](#) (*class in pine.backend.pineiaa.brataia.agree*), 19
[files](#) (*pine.client.CollectionBuilder attribute*), 63
[files](#) (*pine.client.models.CollectionBuilder attribute*), 52
[find_all\(\)](#) (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER method*), 74
[find_all_init\(\)](#) (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER method*), 74
[fit\(\)](#) (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER method*), 73
[fit\(\)](#) (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER method*), 74
[fit\(\)](#) (*pine.pipelines.pipeline.Pipeline method*), 74
[fit\(\)](#) (*pine.pipelines.pmap_ner.NER method*), 75
[fit\(\)](#) (*pine.pipelines.spacy_NER_pipeline.spacy_NER method*), 76
[from_json\(\)](#) (*in module pine.backend.pineiaa.brataia.iaa_service*), 21

flask_get_can_manage_users() (in module *pine.backend.auth.bp*), 5
 flask_get_flat() (in module *pine.backend.auth.bp*), 5
 flask_get_logged_in_user() (in module *pine.backend.auth.bp*), 5
 flask_get_logged_in_user_details() (in module *pine.backend.auth.bp*), 5
 flask_get_login_form() (in module *pine.backend.auth.bp*), 5
 flask_get_module() (in module *pine.backend.auth.bp*), 4
 flask_post_logout() (in module *pine.backend.auth.bp*), 5
 form (*pine.client.CollectionBuilder* attribute), 63
 form (*pine.client.models.CollectionBuilder* attribute), 52
 form_json() (*pine.client.CollectionBuilder* property), 63
 form_json() (*pine.client.models.CollectionBuilder* property), 52
 format_data() (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER_pipeline* method), 73
 format_data() (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER_pipeline* method), 74
 format_data() (*pine.pipelines.spacy_NER_pipeline.spacy_NER_pipeline* method), 76
 from_registration_format() (*pine.pipelines.app.listener.service_listener.ServiceRegistration* class method), 66

G

get() (in module *pine.backend.data.service*), 13
 get() (*pine.client.client.BaseClient* method), 33
 get_all_ids() (*pine.pipelines.EveClient.EveClient* method), 70
 get_all_ids() (*pine.pipelines.NERWrapper.NERWrapper* method), 71
 get_all_items() (in module *pine.backend.pineiaa.bratiaa.iaa_service*), 21
 get_all_items() (*pine.pipelines.EveClient.EveClient* method), 70
 get_all_items() (*pine.pipelines.NERWrapper.NERWrapper* method), 71
 get_all_users() (in module *pine.backend.data.users*), 15
 get_all_users() (*pine.backend.auth.eve.EveModule* method), 6
 get_all_using_pagination() (in module *pine.backend.data.service*), 14
 get_all_versions_of_item_by_id() (in module *pine.backend.data.service*), 14
 get_annotations_for_document() (in module *pine.backend.annotations.bp*), 3
 get_archived_user_collections() (in module *pine.backend.collections.bp*), 9
 get_arg_parser() (*pine.backend.shared.config.ConfigBuilder* class method), 27
 get_arg_parser() (*pine.pipelines.shared.config.ConfigBuilder* class method), 69
 get_auth_module() (*pine.client.client.PineClient* method), 38
 get_auth_module() (*pine.client.PineClient* method), 57
 get_classifier_metrics() (in module *pine.backend.pipelines.bp*), 25
 get_classifier_pipeline_metrics_objs() (*pine.pipelines.NER_API.ner_api* method), 71
 get_collection() (in module *pine.backend.collections.bp*), 10
 get_collection_classifier() (in module *pine.backend.pipelines.bp*), 24
 get_collection_documents() (*pine.client.client.PineClient* method), 39
 get_collection_documents() (*pine.client.PineClient* method), 58
 get_collection_ids_for() (in module *pine.backend.documents.bp*), 15
 get_collection_image() (in module *pine.backend.collections.bp*), 10
 get_collection_image_exists() (in module *pine.backend.collections.bp*), 11
 get_collection_images() (in module *pine.backend.collections.bp*), 10
 get_config() (*pine.backend.shared.config.ConfigBuilder* class method), 27
 get_config() (*pine.pipelines.shared.config.ConfigBuilder* class method), 69
 get_config_names() (*pine.backend.shared.config.ConfigBuilder* static method), 27
 get_config_names() (*pine.pipelines.shared.config.ConfigBuilder* static method), 69
 get_current_annotation() (in module *pine.backend.annotations.bp*), 3
 get_current_report() (in module *pine.backend.pineiaa.bp*), 24
 get_details() (*pine.backend.auth.eve.EveUser* method), 5
 get_doc_and_overlap_ids() (in module *pine.backend.collections.bp*), 10
 get_doc_annotations() (in module *pine.backend.pineiaa.bratiaa.iaa_service*), 21
 get_docs_with_annotations()

(pine.pipelines.EveClient.EveClient method), 70
 get_document() (in module pine.backend.documents.bp), 16
 get_document_ranking() (pine.pipelines.NER_API.ner_api method), 71
 get_documents() (pine.pipelines.EveClient.EveClient method), 70
 get_documents_in_collection() (in module pine.backend.documents.bp), 16
 get_flask_logged_in_user() (in module pine.backend.log), 30
 get_flask_request_info() (in module pine.backend.log), 30
 get_id() (pine.pipelines.corenlp_NER_pipeline.corenlp_NER method), 73
 get_id() (pine.pipelines.opennlp_NER_pipeline.opennlp_NER method), 74
 get_iaa_report_by_collection_id() (in module pine.backend.pineiaa.bp), 24
 get_item_by_id() (in module pine.backend.data.service), 14
 get_items() (in module pine.backend.data.service), 14
 get_items() (in module pine.backend.pineiaa.bratiaa.iaa_service), 21
 get_items() (pine.pipelines.EveClient.EveClient method), 70
 get_items() (pine.pipelines.NERWrapper.NERWrapper method), 71
 get_logged_in_user() (in module pine.backend.auth), 7
 get_logged_in_user() (in module pine.backend.auth.bp), 4
 get_logged_in_user() (pine.backend.auth.bp.AuthModule method), 5
 get_logged_in_user() (pine.client.client.PineClient method), 37
 get_logged_in_user() (pine.client.PineClient method), 56
 get_logged_in_user_details() (pine.backend.auth.bp.AuthModule method), 5
 get_logged_in_user_details() (pine.backend.auth.eve.EveModule method), 6
 get_login_form() (pine.backend.auth.bp.AuthModule method), 5
 get_login_form() (pine.backend.auth.eve.EveModule method), 6
 get_login_form() (pine.backend.auth.oauth.OAuthModule method), 6
 get_login_form_button_text() (pine.backend.auth.oauth.OAuthModule method), 6
 get_login_form_button_text() (pine.backend.auth.vegas.VegasAuthModule method), 7
 get_metrics() (in module pine.backend.pipelines.bp), 24
 get_my_annotations_for_document() (in module pine.backend.annotations.bp), 3
 get_my_user_id() (pine.client.client.PineClient method), 37
 get_my_user_id() (pine.client.PineClient method), 57
 get_next_by_classifier() (in module pine.backend.pipelines.bp), 25
 get_obj() (pine.pipelines.EveClient.EveClient method), 70
 get_obj() (pine.pipelines.NERWrapper.NERWrapper method), 71
 get_others_annotations_for_document() (in module pine.backend.annotations.bp), 3
 get_overlap_ids() (in module pine.backend.collections.bp), 10
 get_pipeline_by_id() (in module pine.backend.pipelines.bp), 24
 get_pipelines() (in module pine.backend.pipelines.bp), 24
 get_pipelines() (pine.client.client.PineClient method), 38
 get_pipelines() (pine.client.PineClient method), 57
 get_registered_channels() (pine.backend.job_manager.service.ServiceManager class method), 17
 get_registered_service_details() (pine.backend.job_manager.service.ServiceManager class method), 17
 get_registered_services() (pine.backend.job_manager.service.ServiceManager class method), 17
 get_resource() (pine.client.client.EveClient method), 35
 get_static_collection_images() (in module pine.backend.collections.bp), 10
 get_unarchived_user_collections() (in module pine.backend.collections.bp), 9
 get_user() (in module pine.backend.admin.bp), 2
 get_user() (in module pine.backend.data.users), 15
 get_user_by_email() (in module pine.backend.data.users), 15
 get_user_can_add_documents_or_images() (in module pine.backend.collections.bp), 11
 get_user_collections() (in module pine.backend.collections.bp), 9
 get_user_details() (in module

pine.backend.data.users), 15
get_users() (in module *pine.backend.admin.bp*), 2
get_users() (*pine.client.client.EveClient* method), 36
getIAAResultForCollection() (in module
pine.backend.pineiaa.bratiaa.iaa_service), 21

H

handle_error() (in module *pine.backend.app*), 28
handle_uncaught_exception() (in module
pine.backend.app), 28
handler_timeout (*pine.backend.job_manager.service.ServiceManager*
attribute), 17
hash_password() (in module
pine.backend.auth.password), 7
hash_password() (in module *pine.client.password*),
55

I

iaa_report() (in module
pine.backend.pineiaa.bratiaa), 22
iaa_report() (in module
pine.backend.pineiaa.bratiaa.agree), 19
id() (*pine.backend.auth.eve.EveUser* property), 5
id() (*pine.backend.auth.oauth.OAuthUser* property), 6
id() (*pine.backend.models.AuthUser* property), 31
ID_FIELD (in module *pine.client.models*), 45
image_file() (*pine.client.CollectionBuilder*
method), 65
image_file() (*pine.client.models.CollectionBuilder*
method), 54
init_app() (in module *pine.backend.admin.bp*), 2
init_app() (in module *pine.backend.annotations.bp*),
4
init_app() (in module *pine.backend.auth.bp*), 5
init_app() (in module *pine.backend.collections.bp*),
11
init_app() (in module *pine.backend.cors*), 29
init_app() (in module *pine.backend.data.bp*), 12
init_app() (in module *pine.backend.documents.bp*),
16
init_app() (in module *pine.backend.pineiaa.bp*), 24
init_app() (in module *pine.backend.pipelines.bp*), 25
init_config() (*pine.backend.shared.config.ConfigBuilder*
class method), 27
init_config() (*pine.pipelines.shared.config.ConfigBuilder*
class method), 69
input_generator() (in module
pine.backend.pineiaa.bratiaa), 23
input_generator() (in module
pine.backend.pineiaa.bratiaa.agree), 19
is_admin() (*pine.backend.auth.eve.EveUser* prop-
erty), 5
is_admin() (*pine.backend.auth.oauth.OAuthUser*
property), 6
is_admin() (*pine.backend.models.AuthUser* prop-
erty), 31
is_cached_last_collection() (in module
pine.backend.collections.bp), 9
is_flat() (in module *pine.backend.auth*), 7
is_flat() (in module *pine.backend.auth.bp*), 4
is_flat() (*pine.backend.auth.bp.AuthModule*
method), 5
is_flat() (*pine.backend.auth.eve.EveModule*
method), 5
is_logged_in() (*pine.backend.auth.oauth.OAuthModule*
method), 6
is_logged_in() (*pine.client.client.PineClient*
method), 38
is_logged_in() (*pine.client.PineClient* method), 57
is_valid() (*pine.client.client.BaseClient* method), 32
is_valid() (*pine.client.client.EveClient* method), 35
is_valid() (*pine.client.client.LocalPineClient*
method), 42
is_valid() (*pine.client.client.PineClient* method), 37
is_valid() (*pine.client.CollectionBuilder* method),
65
is_valid() (*pine.client.LocalPineClient* method), 61
is_valid() (*pine.client.models.CollectionBuilder*
method), 54
is_valid() (*pine.client.PineClient* method), 56
is_valid_annotation() (in module
pine.client.models), 50
is_valid_collection() (in module
pine.client.models), 49
is_valid_doc_annotation() (in module
pine.client.models), 49
is_valid_doc_annotations() (in module
pine.client.models), 50
is_valid_eve_collection() (in module
pine.client.models), 48
is_valid_eve_document() (in module
pine.client.models), 49
is_valid_eve_pipeline() (in module
pine.client.models), 48
is_valid_eve_user() (in module
pine.client.models), 48
is_valid_ner_annotation() (in module
pine.client.models), 50
ITEMS_FIELD (in module *pine.client.models*), 45

L

label() (*pine.client.CollectionBuilder* method), 63
label() (*pine.client.models.CollectionBuilder*
method), 53
labels() (*pine.backend.pineiaa.bratiaa.agree.FIAgreement*
property), 19
labels() (*pine.backend.pineiaa.bratiaa.FIAgreement*
property), 22

largest_margin() (in module *pine.pipelines.RankingFunctions*), 72
 LAST_COLLECTION_FOR_IMAGE (in module *pine.backend.collections.bp*), 9
 least_confidence() (in module *pine.pipelines.RankingFunctions*), 72
 least_confidence_squared() (in module *pine.pipelines.RankingFunctions*), 72
 least_confidence_squared_by_entity() (in module *pine.pipelines.RankingFunctions*), 72
 list_collections() (*pine.client.client.PineClient* method), 41
 list_collections() (*pine.client.PineClient* method), 60
 listener_poll (*pine.pipelines.app.listener.service_listener* attribute), 66
 load_classifier() (*pine.pipelines.NERWrapper.NERWrapper* method), 71
 load_model() (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER* method), 73
 load_model() (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER* method), 74
 load_model() (*pine.pipelines.pipeline.Pipeline* method), 75
 load_model() (*pine.pipelines.pmap_ner.NER* method), 75
 load_model() (*pine.pipelines.spacy_NER_pipeline.spacy_NER* method), 76
 LocalPineClient (class in *pine.client*), 61
 LocalPineClient (class in *pine.client.client*), 42
 logger (in module *pine.backend.annotations.bp*), 3
 LOGGER (in module *pine.backend.app*), 28
 LOGGER (in module *pine.backend.collections.bp*), 9
 logger (in module *pine.backend.data.service*), 12
 logger (in module *pine.backend.job_manager.service*), 16
 logger (in module *pine.backend.pineiaa.bp*), 23
 LOGGER (in module *pine.backend.pineiaa.brataia.agree*), 18
 logger (in module *pine.backend.pipelines.bp*), 24
 LOGGER (in module *pine.backend.shared.config*), 25
 logger (in module *pine.pipelines.app.listener.service_listener*), 66
 logger (in module *pine.pipelines.corenlp_NER_pipeline*), 72
 logger (in module *pine.pipelines.EveClient*), 70
 logger (in module *pine.pipelines.NER_API*), 71
 logger (in module *pine.pipelines.opennlp_NER_pipeline*), 73
 logger (in module *pine.pipelines.pmap_ner*), 75
 LOGGER (in module *pine.pipelines.shared.config*), 67
 LOGGER_DIR (*pine.backend.shared.config.BaseConfig* attribute), 25
 LOGGER_DIR (*pine.pipelines.shared.config.BaseConfig* attribute), 68
 LOGGER_FILE (*pine.backend.shared.config.BaseConfig* attribute), 25
 LOGGER_FILE (*pine.pipelines.shared.config.BaseConfig* attribute), 68
 LOGGER_LEVEL (*pine.backend.shared.config.BaseConfig* attribute), 25
 LOGGER_LEVEL (*pine.pipelines.shared.config.BaseConfig* attribute), 68
 LOGGER_NAME (*pine.backend.shared.config.BaseConfig* attribute), 25
 LOGGER_NAME (*pine.pipelines.shared.config.BaseConfig* attribute), 68
 LOG_SESSION (*pine.backend.log.Action* attribute), 29
 login() (*pine.backend.auth.eve.EveModule* method), 6
 login() (*pine.backend.auth.oauth.OAuthModule* method), 6
 login_eve() (*pine.client.client.PineClient* method), 57
 login_eve() (*pine.client.PineClient* method), 57
 login_required() (in module *pine.backend.auth*), 7
 login_required() (in module *pine.backend.auth.bp*), 4
 LoginForm (class in *pine.backend.models*), 31
 LoginFormField (class in *pine.backend.models*), 30
 LoginFormFieldType (class in *pine.backend.models*), 30
 LOGOUT (*pine.backend.log.Action* attribute), 29
 logout() (*pine.backend.auth.bp.AuthModule* method), 5
 logout() (*pine.client.client.PineClient* method), 38
 logout() (*pine.client.PineClient* method), 57

M

main() (in module *pine.backend.pineiaa.brataia.agree_cli*), 20
 mean_sd_per_document() (*pine.backend.pineiaa.brataia.agree.FIAgreement* method), 19
 mean_sd_per_document() (*pine.backend.pineiaa.brataia.FIAgreement* method), 22
 mean_sd_per_label() (*pine.backend.pineiaa.brataia.agree.FIAgreement* method), 19
 mean_sd_per_label() (*pine.backend.pineiaa.brataia.FIAgreement* method), 22
 mean_sd_per_label_one_vs_rest() (*pine.backend.pineiaa.brataia.agree.FIAgreement* method), 19
 mean_sd_per_label_one_vs_rest() (*pine.backend.pineiaa.brataia.FIAgreement* method), 19

method), 23
 mean_sd_total() (*pine.backend.pineiaa.bratiaa.agree.FIAgreement*
method), 19
 mean_sd_total() (*pine.backend.pineiaa.bratiaa.FIAgreement*
method), 22
 mean_sd_total_one_vs_rest()
 (*pine.backend.pineiaa.bratiaa.agree.FIAgreement*
method), 19
 mean_sd_total_one_vs_rest()
 (*pine.backend.pineiaa.bratiaa.FIAgreement*
method), 23
 message (*pine.client.exceptions.PineClientException*
attribute), 43
 metadata() (*pine.client.CollectionBuilder* *method*),
 63
 metadata() (*pine.client.models.CollectionBuilder*
method), 53
 MODELS_DIR (*pine.pipelines.shared.config.BaseConfig*
attribute), 68
 MODELS_LOCAL_DIR (*pine.backend.shared.config.BaseConfig*
attribute), 26
 module
 pine, 1
 pine.backend, 1
 pine.backend.admin, 1
 pine.backend.admin.bp, 1
 pine.backend.annotations, 2
 pine.backend.annotations.bp, 2
 pine.backend.app, 28
 pine.backend.auth, 4
 pine.backend.auth.bp, 4
 pine.backend.auth.eve, 5
 pine.backend.auth.oauth, 6
 pine.backend.auth.password, 7
 pine.backend.auth.vegas, 7
 pine.backend.collections, 8
 pine.backend.collections.bp, 8
 pine.backend.config, 28
 pine.backend.cors, 28
 pine.backend.data, 12
 pine.backend.data.bp, 12
 pine.backend.data.service, 12
 pine.backend.data.users, 14
 pine.backend.documents, 15
 pine.backend.documents.bp, 15
 pine.backend.job_manager, 16
 pine.backend.job_manager.service, 16
 pine.backend.log, 29
 pine.backend.models, 30
 pine.backend.pineiaa, 18
 pine.backend.pineiaa.bp, 23
 pine.backend.pineiaa.bratiaa, 18
 pine.backend.pineiaa.bratiaa.agree,
 18
 pine.backend.pineiaa.bratiaa.agree_cli,
 20
 pine.backend.pineiaa.bratiaa.evaluation,
 20
 pine.backend.pineiaa.bratiaa.iaa_service,
 21
 pine.backend.pineiaa.bratiaa.utils,
 21
 pine.backend.pipelines, 24
 pine.backend.pipelines.bp, 24
 pine.backend.shared, 25
 pine.backend.shared.config, 25
 pine.backend.shared.transform, 27
 pine.client, 31
 pine.client.client, 31
 pine.client.exceptions, 43
 pine.client.log, 44
 pine.client.models, 44
 pine.client.password, 55
 pine.pipelines, 65
 pine.pipelines.app, 65
 pine.pipelines.app.listener, 65
 pine.pipelines.app.listener.main, 65
 pine.pipelines.app.listener.service_listener,
 66
 pine.pipelines.corenlp_NER_pipeline,
 72
 pine.pipelines.EveClient, 70
 pine.pipelines.NER_API, 71
 pine.pipelines.NERWrapper, 71
 pine.pipelines.opennlp_NER_pipeline,
 73
 pine.pipelines.pipeline, 74
 pine.pipelines.pmap_ner, 75
 pine.pipelines.RankingFunctions, 72
 pine.pipelines.run_service, 75
 pine.pipelines.shared, 67
 pine.pipelines.shared.config, 67
 pine.pipelines.shared.transform, 70
 pine.pipelines.spacy_NER_pipeline,
 75
 module (*in module pine.backend.auth*), 7
 module (*in module pine.backend.auth.bp*), 4
 mongo (*pine.client.client.EveClient* *attribute*), 35
 mongo_base_uri (*pine.client.client.EveClient* *at-*
 tribute), 35
 mongo_db (*pine.client.client.EveClient* *attribute*), 35
 most_of_least_popular() (*in module*
 pine.pipelines.RankingFunctions), 72
 N
 NER (*class in pine.pipelines.pmap_ner*), 75
 ner_api (*class in pine.pipelines.NER_API*), 71
 NERWrapper (*class in pine.pipelines.NERWrapper*), 71

`next_example()` (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER*
method), 73 *pine.backend.app*
module, 28
`next_example()` (*pine.pipelines.opennlp_NER_pipeline.opennlp_NER*
method), 74 *pine.backend.auth*
module, 4
`next_example()` (*pine.pipelines.pipeline.Pipeline*
method), 75 *pine.backend.auth.bp*
module, 4
`next_example()` (*pine.pipelines.pmap_ner.NER*
method), 75 *pine.backend.auth.eve*
module, 5
`next_example()` (*pine.pipelines.spacy_NER_pipeline.spacy_NER*
method), 76 *pine.backend.auth.oauth*
module, 6
`not_found()` (*in module pine.backend.cors*), 28 *pine.backend.auth.password*
module, 7
pine.backend.auth.vegas
module, 7
pine.backend.collections
module, 8
pine.backend.collections.bp
module, 8
pine.backend.config
module, 28
pine.backend.cors
module, 28
pine.backend.data
module, 12
pine.backend.data.bp
module, 12
pine.backend.data.service
module, 12
pine.backend.data.users
module, 14
pine.backend.documents
module, 15
pine.backend.documents.bp
module, 15
pine.backend.job_manager
module, 16
pine.backend.job_manager.service
module, 16
pine.backend.log
module, 29
pine.backend.models
module, 30
pine.backend.pineiaa
module, 18
pine.backend.pineiaa.bp
module, 23
pine.backend.pineiaa.bratiaa
module, 18
pine.backend.pineiaa.bratiaa.agree
module, 18
pine.backend.pineiaa.bratiaa.agree_cli
module, 20
pine.backend.pineiaa.bratiaa.evaluation
module, 20

O

OAuthModule (*class in pine.backend.auth.oauth*), 6
OAuthUser (*class in pine.backend.auth.oauth*), 6
opennlp_NER (*class in pine.pipelines.opennlp_NER_pipeline*), 73
overlapping_tokens() (*pine.backend.pineiaa.bratiaa.utils.TokenOverlap*
method), 22

P

`params()` (*in module pine.backend.data.service*), 13
`parse_args()` (*in module pine.backend.pineiaa.bratiaa.agree_cli*),
20
`parser` (*in module pine.pipelines.NERWrapper*), 71
PASSWORD (*pine.backend.models.LoginFormFieldType*
attribute), 30
`patch()` (*in module pine.backend.data.service*), 14
`patch()` (*pine.client.client.BaseClient method*), 34
`perform_five_fold()` (*pine.pipelines.NER_API.ner_api method*),
71
`perform_fold()` (*pine.pipelines.NER_API.ner_api method*), 71
PERFORMANCE_HISTORY (*in module pine.backend.data.service*), 13
PerformanceHistory (*class in pine.backend.data.service*), 12
`pformat()` (*pine.backend.data.service.PerformanceHistory*
method), 13
pine
module, 1
pine.backend
module, 1
pine.backend.admin
module, 1
pine.backend.admin.bp
module, 1
pine.backend.annotations
module, 2
pine.backend.annotations.bp
module, 2

pine.backend.pineiaa.bratiaa.iaa_service
 module, 21
 pine.backend.pineiaa.bratiaa.utils
 module, 21
 pine.backend.pipelines
 module, 24
 pine.backend.pipelines.bp
 module, 24
 pine.backend.shared
 module, 25
 pine.backend.shared.config
 module, 25
 pine.backend.shared.transform
 module, 27
 pine.client
 module, 31
 pine.client.client
 module, 31
 pine.client.exceptions
 module, 43
 pine.client.log
 module, 44
 pine.client.models
 module, 44
 pine.client.password
 module, 55
 pine.pipelines
 module, 65
 pine.pipelines.app
 module, 65
 pine.pipelines.app.listener
 module, 65
 pine.pipelines.app.listener.main
 module, 65
 pine.pipelines.app.listener.service_listener
 module, 66
 pine.pipelines.corenlp_NER_pipeline
 module, 72
 pine.pipelines.EveClient
 module, 70
 pine.pipelines.NER_API
 module, 71
 pine.pipelines.NERWrapper
 module, 71
 pine.pipelines.opennlp_NER_pipeline
 module, 73
 pine.pipelines.pipeline
 module, 74
 pine.pipelines.pmap_ner
 module, 75
 pine.pipelines.RankingFunctions
 module, 72
 pine.pipelines.run_service
 module, 75
 pine.pipelines.shared
 module, 67
 pine.pipelines.shared.config
 module, 67
 pine.pipelines.shared.transform
 module, 70
 pine.pipelines.spacy_NER_pipeline
 module, 75
 PineClient (class in pine.client), 56
 PineClient (class in pine.client.client), 37
 PineClientAuthException, 43
 PineClientException, 43
 PineClientHttpException, 43
 PineClientValueException, 43
 ping () (pine.client.client.EveClient method), 35
 ping () (pine.client.client.PineClient method), 37
 ping () (pine.client.PineClient method), 56
 pipe_init () (pine.pipelines.pmap_ner.NER method), 75
 Pipeline (class in pine.pipelines.pipeline), 74
 pipeline (pine.pipelines.pmap_ner.NER attribute), 75
 PIPELINE (pine.pipelines.shared.config.BaseConfig attribute), 68
 post () (in module pine.backend.data.service), 13
 post () (pine.client.client.BaseClient method), 34
 post_collection_image () (in module pine.backend.collections.bp), 11
 pprint () (pine.backend.data.service.PerformanceHistory method), 13
 pre_process_message ()
 (pine.pipelines.app.listener.service_listener.ServiceListener method), 67
 predict () (in module pine.backend.pipelines.bp), 25
 predict () (pine.pipelines.corenlp_NER_pipeline.corenlp_NER method), 73
 predict () (pine.pipelines.NER_API.ner_api method), 72
 predict () (pine.pipelines.NERWrapper.NERWrapper method), 71
 predict () (pine.pipelines.opennlp_NER_pipeline.opennlp_NER method), 74
 predict () (pine.pipelines.pipeline.Pipeline method), 74
 predict () (pine.pipelines.pmap_ner.NER method), 75
 predict () (pine.pipelines.spacy_NER_pipeline.spacy_NER method), 76
 predict_proba () (pine.pipelines.corenlp_NER_pipeline.corenlp_NER method), 73
 predict_proba () (pine.pipelines.opennlp_NER_pipeline.opennlp_NER method), 74
 predict_proba () (pine.pipelines.pipeline.Pipeline method), 75
 predict_proba () (pine.pipelines.pmap_ner.NER method), 75

[predict_proba\(\)](#) ([pine.pipelines.spacy_NER_pipeline.spacy_NER](#) [pine.backend.job_manager.service.ServiceManager](#) [method](#)), 76 [attribute](#)), 16
[preprocessing_lock_key](#) [r_pool](#) ([pine.pipelines.app.listener.service_listener.ServiceListener](#) [attribute](#)), 66
[preprocessing_lock_key_timeout](#) [random_rank\(\)](#) (in [module](#) [pine.pipelines.RankingFunctions](#)), 72
[preprocessing_worker_lock_key](#) [read\(\)](#) (in [module](#) [pine.backend.pineiaa.brat1aa.utils](#)), 21
[preprocessing_worker_lock_key_timeout](#) [redis_channel_ttl_key_prefix](#) ([pine.backend.job_manager.service.ServiceManager](#) [attribute](#)), 17
[print_table\(\)](#) ([pine.backend.pineiaa.brat1aa.agree.FIAgreement](#) [pine.backend.job_manager.service.ServiceManager](#) [static method](#)), 19 [attribute](#)), 16
[print_table\(\)](#) ([pine.backend.pineiaa.brat1aa.FIAgreement](#) [redis_channels_key_ttl](#) ([pine.backend.job_manager.service.ServiceManager](#) [attribute](#)), 17
[print_users_command\(\)](#) (in [module](#) [pine.backend.data.users](#)), 15 [attribute](#)), 17
[process_message\(\)](#) [REDIS_DBNUM](#) ([pine.backend.shared.config.BaseConfig](#) [attribute](#)), 26
[processing_limit](#) ([pine.pipelines.app.listener.service_listener.ServiceListener](#) [attribute](#)), 67
[processing_lock_key](#) [REDIS_EXPIRE](#) ([pine.pipelines.shared.config.BaseConfig](#) [attribute](#)), 68
[processing_lock_key_timeout](#) [redis_handler_mutex_key_prefix](#) ([pine.backend.job_manager.service.ServiceManager](#) [attribute](#)), 17
[processing_queue_key](#) [redis_handler_mutex_key_ttl](#) ([pine.backend.job_manager.service.ServiceManager](#) [attribute](#)), 17
[processing_queue_key_timeout](#) [REDIS_HOST](#) ([pine.backend.shared.config.BaseConfig](#) [attribute](#)), 26
[processing_worker_name](#) [REDIS_HOST](#) ([pine.pipelines.shared.config.BaseConfig](#) [attribute](#)), 68
[processor_poll](#) ([pine.pipelines.app.listener.service_listener.ServiceListener](#) [attribute](#)), 67
[publish_response\(\)](#) [redis_key_prefix](#) ([pine.backend.job_manager.service.ServiceManager](#) [attribute](#)), 16
[put\(\)](#) (in [module](#) [pine.backend.data.service](#)), 13 [REDIS_PORT](#) (in [module](#) [pine.backend.config](#)), 28
[put\(\)](#) ([pine.client.client.BaseClient](#) [method](#)), 33 [REDIS_PORT](#) ([pine.pipelines.shared.config.BaseConfig](#) [attribute](#)), 26
[R](#) [REDIS_PREFIX](#) ([pine.backend.shared.config.BaseConfig](#) [attribute](#)), 26
[r_conn](#) ([pine.backend.job_manager.service.ServiceManager](#) [attribute](#)), 16 [REDIS_PWD](#) ([pine.pipelines.shared.config.BaseConfig](#) [attribute](#)), 68
[r_conn](#) ([pine.pipelines.app.listener.service_listener.ServiceListener](#) [attribute](#)), 66 [redis_reg_key_prefix](#) ([pine.backend.job_manager.service.ServiceManager](#) [attribute](#)), 66

attribute), 16
 redis_reg_key_ttl (pine.backend.job_manager.service.ServiceManager attribute), 17
 REDIS_SERVER (in module pine.backend.config), 28
 REDIS_USR (pine.backend.shared.config.BaseConfig attribute), 26
 REDIS_USR (pine.pipelines.shared.config.BaseConfig attribute), 68
 redis_work_mutex_key_prefix (pine.backend.job_manager.service.ServiceManager attribute), 17
 redis_work_mutex_key_ttl (pine.backend.job_manager.service.ServiceManager attribute), 17
 redis_work_queue_key_prefix (pine.backend.job_manager.service.ServiceManager attribute), 17
 redis_work_queue_key_ttl (pine.backend.job_manager.service.ServiceManager attribute), 17
 register_oauth() (pine.backend.auth.oauth.OAuthModule method), 6
 register_oauth() (pine.backend.auth.vegas.VegasAuthModule method), 7
 registration_channel (pine.backend.job_manager.service.ServiceManager attribute), 17
 registration_channel (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 67
 registration_poll (pine.pipelines.app.listener.service_listener.ServiceListener attribute), 66
 registration_worker_name (pine.backend.job_manager.service.ServiceManager attribute), 17
 remove_eve_fields() (in module pine.backend.data.service), 14
 remove_eve_fields() (in module pine.client.models), 54
 remove_nonupdatable_fields() (in module pine.backend.data.service), 14
 remove_nonupdatable_fields() (in module pine.client.models), 55
 reserved_channels (pine.backend.job_manager.service.ServiceManager attribute), 17
 reset_user_passwords() (in module pine.backend.data.users), 15
 resp (pine.client.exceptions.PineClientHttpException attribute), 43
 ROOT_DIR (pine.backend.shared.config.BaseConfig attribute), 25
 ROOT_DIR (pine.pipelines.shared.config.BaseConfig attribute), 68
 run() (in module pine.pipelines.app.listener.main), 66
 S
 save_annotations() (in module pine.backend.annotations.bp), 4
 save_collection_annotations() (in module pine.backend.annotations.bp), 4
 save_model() (pine.pipelines.corenlp_NER_pipeline.corenlp_NER method), 73
 save_model() (pine.pipelines.opennlp_NER_pipeline.opennlp_NER method), 74
 save_model() (pine.pipelines.pipeline.Pipeline method), 75
 save_model() (pine.pipelines.pmap_ner.NER method), 75
 save_model() (pine.pipelines.spacy_NER_pipeline.spacy_NER method), 76
 SCHEDULER_HANDLER_TIMEOUT (pine.backend.shared.config.BaseConfig attribute), 26
 SCHEDULER_HANDLER_TIMEOUT (pine.pipelines.shared.config.BaseConfig attribute), 68
 SCHEDULER_QUEUE_TIMEOUT (pine.backend.shared.config.BaseConfig attribute), 26
 SCHEDULER_QUEUE_TIMEOUT (pine.pipelines.shared.config.BaseConfig attribute), 68
 SCHEDULER_REGISTRATION_TIMEOUT (pine.backend.shared.config.BaseConfig attribute), 26
 SCHEDULER_REGISTRATION_TIMEOUT (pine.pipelines.shared.config.BaseConfig attribute), 68
 SECRET_KEY (in module pine.backend.config), 28
 send_service_request() (pine.backend.job_manager.service.ServiceManager class method), 17
 SERVICE_HANDLER_TIMEOUT (pine.backend.shared.config.BaseConfig attribute), 26
 SERVICE_HANDLER_TIMEOUT (pine.pipelines.shared.config.BaseConfig attribute), 68
 SERVICE_LIST (pine.backend.shared.config.BaseConfig attribute), 26
 SERVICE_LIST (pine.pipelines.shared.config.BaseConfig attribute), 68
 SERVICE_LISTENING_FREQUENCY (pine.backend.shared.config.BaseConfig attribute), 26

SERVICE_LISTENING_FREQUENCY
 (*pine.pipelines.shared.config.BaseConfig*
 attribute), 68
 service_manager (in module
 pine.backend.pipelines.bp), 24
 SERVICE_REGISTRATION_CHANNEL
 (*pine.backend.shared.config.BaseConfig*
 attribute), 26
 SERVICE_REGISTRATION_CHANNEL
 (*pine.pipelines.shared.config.BaseConfig*
 attribute), 68
 SERVICE_REGISTRATION_FREQUENCY
 (*pine.backend.shared.config.BaseConfig*
 attribute), 26
 SERVICE_REGISTRATION_FREQUENCY
 (*pine.pipelines.shared.config.BaseConfig*
 attribute), 68
 ServiceListener (class in
 pine.pipelines.app.listener.service_listener), 66
 ServiceManager (class in
 pine.backend.job_manager.service), 16
 ServiceRegistration (class in
 pine.pipelines.app.listener.service_listener), 66
 session (*pine.client.client.BaseClient* *attribute*), 32
 set_config() (*pine.backend.shared.config.ConfigBuilder*
 class method), 27
 set_config() (*pine.pipelines.shared.config.ConfigBuilder*
 class method), 69
 set_document_to_annotated_by_user() (in
 module *pine.backend.annotations.bp*), 3
 set_user_password() (in module
 pine.backend.data.users), 15
 set_user_password_by_id() (in module
 pine.backend.data.users), 15
 setup_logging() (in module *pine.backend.log*), 30
 setup_logging() (in module *pine.client*), 61
 setup_logging() (in module *pine.client.log*), 44
 setup_logging() (in module
 pine.pipelines.app.listener.main), 66
 shutdown_channel (*pine.backend.job_manager.service.ServiceManager*
 attribute), 17
 sm (in module *pine.backend.job_manager.service*), 18
 spacy_NER (class in
 pine.pipelines.spacy_NER_pipeline), 76
 start_listeners()
 (*pine.backend.job_manager.service.ServiceManager*
 method), 17
 start_workers() (*pine.pipelines.app.listener.service_listener.ServiceListener*
 method), 67
 stop_listeners() (*pine.backend.job_manager.service.ServiceManager*
 method), 17
 stop_workers() (*pine.pipelines.app.listener.service_listener.ServiceListener*
 method), 67
 system_export() (in module
 pine.backend.admin.bp), 2
 system_import() (in module
 pine.backend.admin.bp), 2
T
 test_redis() (in module *pine.backend.pipelines.bp*),
 25
 TestConfig (class in *pine.backend.shared.config*), 26
 TestConfig (class in *pine.pipelines.shared.config*), 69
 TESTING (*pine.backend.shared.config.BaseConfig* *at-*
 tribute), 25
 TESTING (*pine.pipelines.shared.config.BaseConfig* *at-*
 tribute), 68
 TEXT (*pine.backend.models.LoginFormFieldType* *at-*
 tribute), 30
 title() (*pine.client.CollectionBuilder* method), 64
 title() (*pine.client.models.CollectionBuilder*
 method), 53
 to_dict() (*pine.backend.models.AuthUser* method),
 31
 to_dict() (*pine.backend.models.LoginForm* method),
 31
 to_dict() (*pine.backend.models.LoginFormField*
 method), 30
 to_dict() (*pine.backend.models.UserDetails*
 method), 31
 to_registration_format()
 (*pine.pipelines.app.listener.service_listener.ServiceRegistration*
 method), 66
 TOKEN (in module *pine.backend.pineiaa.brataiaa.utils*),
 21
 tokenize() (in module *pine.backend.pineiaa.brataiaa*),
 23
 tokenize() (in module
 pine.backend.pineiaa.brataiaa.utils), 21
 tokenize() (*pine.pipelines.corenlp_NER_pipeline.corenlp_NER*
 method), 73
 TokenOverlap (class in
 pine.backend.pineiaa.brataiaa.utils), 21
 transform_module_by_config() (*pine.pipelines.NER_API.ner_api*
 method), 71
 transform_module_by_config() (in module
 pine.backend.shared.transform), 27
 transform_module_by_config() (in module
 pine.pipelines.shared.transform), 70
U
 unarchive_collection() (in module
 pine.backend.collections.bp), 10
 update() (*pine.pipelines.EveClient.EveClient*
 method), 70
 update() (*pine.pipelines.NERWrapper.NERWrapper*
 method), 71

update_cached_last_collection() (in module *pine.backend.collections.bp*), 9
 update_metadata() (in module *pine.backend.documents.bp*), 16
 update_model() (*pine.pipelines.NERWrapper.NERWrapper* method), 71
 update_user() (in module *pine.backend.admin.bp*), 2
 update_user() (in module *pine.backend.data.users*), 15
 update_user_details() (*pine.backend.auth.eve.EveModule* method), 6
 update_user_password() (in module *pine.backend.admin.bp*), 2
 update_user_password() (*pine.backend.auth.eve.EveModule* method), 6
 uri() (*pine.client.client.BaseClient* method), 33
 url() (in module *pine.backend.data.service*), 13
 user_can_add_documents_or_images() (in module *pine.backend.collections*), 11
 user_can_add_documents_or_images() (in module *pine.backend.collections.bp*), 9
 user_can_add_documents_or_images_by_id() (in module *pine.backend.collections*), 11
 user_can_add_documents_or_images_by_id() (in module *pine.backend.collections.bp*), 9
 user_can_annotate() (in module *pine.backend.collections*), 11
 user_can_annotate() (in module *pine.backend.collections.bp*), 9
 user_can_annotate() (in module *pine.backend.documents.bp*), 16
 user_can_annotate_by_id() (in module *pine.backend.collections*), 11
 user_can_annotate_by_id() (in module *pine.backend.collections.bp*), 9
 user_can_annotate_by_id() (in module *pine.backend.documents.bp*), 16
 user_can_annotate_by_ids() (in module *pine.backend.collections*), 11
 user_can_annotate_by_ids() (in module *pine.backend.collections.bp*), 9
 user_can_annotate_by_ids() (in module *pine.backend.documents.bp*), 16
 user_can_modify_document_metadata() (in module *pine.backend.collections*), 11
 user_can_modify_document_metadata() (in module *pine.backend.collections.bp*), 9
 user_can_modify_document_metadata_by_id() (in module *pine.backend.collections*), 11
 user_can_modify_document_metadata_by_id() (in module *pine.backend.collections.bp*), 9
 user_can_modify_metadata() (in module *pine.backend.documents.bp*), 16
 user_can_modify_metadata_by_id() (in module *pine.backend.documents.bp*), 16
 user_can_view() (in module *pine.backend.collections*), 11
 user_can_view() (in module *pine.backend.collections.bp*), 9
 user_can_view() (in module *pine.backend.documents.bp*), 16
 user_can_view_by_id() (in module *pine.backend.collections*), 11
 user_can_view_by_id() (in module *pine.backend.collections.bp*), 9
 user_can_view_by_id() (in module *pine.backend.documents.bp*), 16
 UserDetails (class in *pine.backend.models*), 31
 username() (*pine.backend.auth.eve.EveUser* property), 5
 username() (*pine.backend.auth.oauth.OAuthUser* property), 6
 username() (*pine.backend.models.AuthUser* property), 31

V

VEGAS_CLIENT_SECRET (in module *pine.backend.config*), 28
 VegasAuthModule (class in *pine.backend.auth.vegas*), 7
 VERSION (in module *pine.backend*), 31
 VERSION (in module *pine.backend.app*), 28
 VIEW_DOCUMENT (*pine.backend.log.Action* attribute), 29
 viewer() (*pine.client.CollectionBuilder* method), 63
 viewer() (*pine.client.models.CollectionBuilder* method), 52

W

where_params() (in module *pine.backend.data.service*), 13