
cv_kickstarter Documentation

Release

Author

December 04, 2014

CONTENTS

1	cv_kickstarter package	3
1.1	Subpackages	3
1.2	Submodules	4
1.3	cv_kickstarter.academic_skill_set module	4
1.4	cv_kickstarter.cnapi module	6
1.5	cv_kickstarter.course_keyword_tokenizer module	9
1.6	cv_kickstarter.course_repository module	10
1.7	cv_kickstarter.cv_kickstarter_config module	11
1.8	cv_kickstarter.dtu_course_base module	11
1.9	cv_kickstarter.dtu_skill_set module	12
1.10	cv_kickstarter.ects_grade_calculator module	13
1.11	cv_kickstarter.nltk_data_downloader module	13
1.12	cv_kickstarter.session_authentication module	13
1.13	Module contents	14
2	job_searcher package	15
2.1	Submodules	15
2.2	job_searcher.career_builder module	15
2.3	job_searcher.go_jobs module	15
2.4	job_searcher.job module	16
2.5	job_searcher.keyword_evaluator module	16
2.6	Module contents	17
3	Using CV Kickstarter	19
3.1	Setup	19
3.2	Configuration	19
3.3	Run webserver	19
3.4	Commmand Line Integration	19
3.5	Tests	20
4	Indices and tables	21
	Python Module Index	23
	Index	25

Contents:

CV_KICKSTARTER PACKAGE

1.1 Subpackages

1.1.1 cv_kickstarter.models package

Submodules

cv_kickstarter.models.exam_result_programme module

The exam result programme with exam results for a given programme.

```
class cv_kickstarter.models.exam_result_programme.ExamResultProgramme(name,
                                                                    passed_ects,
                                                                    to-
                                                                    tal_ects,
                                                                    exam_results)
```

Bases: object

Representing exam results for a given programme (bachelor, master..).

average_grade

Return the average grade of the student.

is_done

Return true if all the courses are passed for the programme.

cv_kickstarter.models.user_cv module

UserCV for displaying the students CV on the frontend.

```
class cv_kickstarter.models.user_cv.UserCV(first_name, last_name, exam_result_programmes,
                                           keywords)
```

Bases: object

UserCV object works as an view object to be used in the view.

course_title_sentence (keyword)

Return a course title sentence.

If one course: 01234 Course For two courses: 01234 Course 1 and 01235 Course 2 For three courses:
01234 C1, 01235 C2 and 012346 C3

full_name

Return the full name of the student.

highest_ranked_keywords

Return the 50 highest ranked keywords.

cv_kickstarter.models.user_cv_builder module

Builds a User CV to display on the frontend.

class cv_kickstarter.models.user_cv_builder.**CampusNetExamResultMapper** (*cn_exam_result_programme*)
Bases: object

Maps an exam result programme to an object ready for frontend.

mapped_exam_result ()
Return mapped exam result.

class cv_kickstarter.models.user_cv_builder.**UserCVBuilder** (*campus_net_client*,
mongo_store)

Bases: object

Builds a User CV to display on the frontend.

build ()
Return a UserCV.

grades
Return grades from campus_net_client.

user
Return a user with information from the campus_net_client.

cv_kickstarter.models.user_cv_dictionary_mapper module

Maps a UserCV into a dictionary.

class cv_kickstarter.models.user_cv_dictionary_mapper.**UserCVDictionaryMapper**
Bases: object

Maps a UserCV into a dictionary.

user_cv_dict (*user_cv*, *student_number*)
Return a dictionary based on the UserCV.

Module contents

The domain specific code for this project.

1.2 Submodules

1.3 cv_kickstarter.academic_skill_set module

Extracts a skill set based on keywords in exam results.

AcademicSkillSet extracts a skill set based on keywords in the exam results.

The keywords are ranked by the frequency and grades.

```

class cv_kickstarter.academic_skill_set.CourseKeyword
    Bases: tuple

    CourseKeyword(keyword, rank, course_numbers)

    course_numbers
        Alias for field number 2

    keyword
        Alias for field number 0

    rank
        Alias for field number 1

class cv_kickstarter.academic_skill_set.CourseSkillSet (grade_booster, word_scorer)
    Bases: object

    Extract skill gained in a given course.

    skill_set (tokenized_exam_result)
        Return a skill set gained in the given course.

class cv_kickstarter.academic_skill_set.CourseSkillSetMerger
    Bases: object

    Merges skill sets from several courses into a common skill set.

    student_skill_set (passed_courses_skills)
        Return a raw skill set merged.

        Each course skill set is merged, which means that common skills from the different courses are merged
        into one CourseKeyword, which lists both courses.

        For keywords contained in two course skill sets, the rank is summed as the keyword gained in more courses
        are considered more important.

class cv_kickstarter.academic_skill_set.KeywordGradeBooster (average_grade)
    Bases: object

    Boosts the keyword rank with the grade.

    Multiplies the given keyword rank score with a grade score from the course the course originated.

    boosted_keyword_scores (tokenized_exam_result, scored_keywords)
        Return a list of keywords with a rank boosted by grade.

class cv_kickstarter.academic_skill_set.KeywordScoreCalculator
    Bases: object

    "Calculates the raw keyword scores based on the word scores.

    keyword_scorer (tokenized_exam_result, word_scorer)
        "Return keyword scores.

class cv_kickstarter.academic_skill_set.KeywordScoreNormalizer
    Bases: object

    Normalize the rank score of a keyword with average score in course.

    normalized_keyword_score (keyword_scorer)
        Return a list of keywords with rank divided by average rank.

class cv_kickstarter.academic_skill_set.StudentSkillSet (word_scorer, grade_booster,
                                                         noise_filter)
    Bases: object

```

Extract skill set of a student.

skill_set (*tokenized_exam_results*)
Return a ranked skill set.

class cv_kickstarter.academic_skill_set.**StudentSkillSetNoiceFilter** (*min_keyword_length*)
Bases: object

Filters away noice from the skill set.

Filters away noice meaning unwanted keywords in the skill set.

filtered_skill_set (*course_keywords*)
Return a filtered set of skills.

class cv_kickstarter.academic_skill_set.**WordFrequencyScoreCalculator**
Bases: object

Builds a word score dictionary based on word frequency.

word_scorer (*tokenized_course_exam_result*)
Return a dictionary of word scores.

The score for each word is the frequency of the word in all exam results.

cv_kickstarter.academic_skill_set.**skill_set** (*tokenized_exam_results*,
min_keyword_length=4)

Extract skill set based on tokenized exam results.

Tokenized exam results are exam results with tokens for each course.

Optimal arguments are min_keyword_length: The minimum amount of characters in a keyword

The last max_keyword_courses is used for filtering away keywords that are too common in the courses (e.g. 'course', 'analysis').

1.4 cv_kickstarter.cnapi module

Python client for the CampusNet API.

This library is designed for having a nice interface for integrating with the CampusNet API. The library provides an interface for the network requests as well as objects for wrapping the data returned by the CampusNet API.

For documentation of the Campusnet API, see:

<https://www.campusnet.dtu.dk/data/Documentation/CampusNet%20public%20API.pdf>

Example of usage:

At first instantiate the api:

```
>>> app_name = 'MyCampusNetApp'
>>> app_token = 'sh2870272-2ush292-ji2u98s2-2h2821-jsw9j2ihs982'

>>> api = cnapi.CampusNetApi(app_name, app_token)
```

In order to fetch information, authenticate the student with the student number and password:

```
>>> api.authenticate('s123456', 'secret-password')
```

or if the auth token is already in possession use:

```
>>> api.authenticate_with_token('s123456', '21EF8196-ED05-4BAB-9081')
```

To fetch the grades of the given user:

```
>>> grades = api.grades()
```

To fetch the user infor of the given user:

```
>>> user = api.user()
```

class cv_kickstarter.cnapi.**AbstractXmlInfoExtractor** (*response_text*)

An abstract class for classes that extract information from xml.

The inheriting class needs to implement:

`_extract_information`

extract ()

Extract the information from the given xml.

Returns a structure given by the child class implementing ‘_extract_information’.

Returns None if the CampusNet API returns a Fault

class cv_kickstarter.cnapi.**Authenticator** (*app_name, api_token*)

Can authenticate a user based on a user name and a password.

The authentication fetches an auth_token from the CampusNet API that authenticates the user.

Example usage:

```
>>> auth = Authenticator('MyApp', 'app-token-123')
```

```
>>> token = auth.auth_token('s1234', 'secretpass')
```

auth_token (*username, password*)

Fetch and return an authentication token from CampusNet API.

The request is a POST request send the given app name, app token, username and password.

If the authentication fails, it will return None.

class cv_kickstarter.cnapi.**CampusNetApi** (*app_name, api_token*)

The interface class for the API.

This class acts as the top level interface of the api and wraps the behaviour needed for fetching relevant information from the API.

authenticate (*student_number, password*)

Authenticate the given user by fetching an authentication token.

authenticate_with_token (*student_number, auth_token*)

Authenticate the given user by the given authentication token.

grades ()

Fetch the grades for the authenticated user.

is_authenticated ()

Return a boolean indicating whether the user is authenticated.

user ()

Fetch user infor for the authenticated user.

user_picture (*user_id*)

“Fetch the user picture.

```
class cv_kickstarter.cnapi.ExamResult
    Bases: tuple

    ExamResult(course_title, course_number, ects_points, grade, period, year)

    course_number
        Alias for field number 1

    course_title
        Alias for field number 0

    ects_points
        Alias for field number 2

    grade
        Alias for field number 3

    period
        Alias for field number 4

    year
        Alias for field number 5

class cv_kickstarter.cnapi.ExamResultXmlMapper(exam_result_xml)
    Bases: object

    Is able to extract and map exam result xml into ExamResult objects.

    exam_result ()
        Return an exam result object with information given by the xml.

class cv_kickstarter.cnapi.ProgramExamResults
    Bases: tuple

    ProgramExamResults(name, is_active, passed_ects_points, exam_results)

    exam_results
        Alias for field number 3

    is_active
        Alias for field number 1

    name
        Alias for field number 0

    passed_ects_points
        Alias for field number 2

class cv_kickstarter.cnapi.Student
    Bases: tuple

    Student(first_name, last_name, email, user_id)

    email
        Alias for field number 2

    first_name
        Alias for field number 0

    last_name
        Alias for field number 1

    user_id
        Alias for field number 3
```

class `cv_kickstarter.cnapi.UserClient` (*app_name, api_token, student_number, access_token*)
 Network client for fetching user information CampusNet API.

get (*path*)

Perform a GET request to fetch information about the given user.

For example:

```
user_client = UserClient('MyApp', 'api-token-123', 's123', 'atoken') user_client.get('Grades')
```

will perform a GET request to:

```
https://www.campusnet.dtu.dk/data/CurrentUser/Grades
```

with the headers:

```
X-appname: 'MyApp' X-token: 'api-token-123' accept-language: 'da-DK' X-Include-services-
and-relations: 'true'
```

class `cv_kickstarter.cnapi.UserGradesExtractor` (*response_text*)
 Bases: `cv_kickstarter.cnapi.AbstractXmlInfoExtractor`

Is able to extract the grades of the given user.

class `cv_kickstarter.cnapi.UserInfoExtractor` (*response_text*)
 Bases: `cv_kickstarter.cnapi.AbstractXmlInfoExtractor`

Is able to extract user info based on xml describe the user.

1.5 cv_kickstarter.course_keyword_tokenizer module

Extract raw keywords from course.

`course_keyword_tokenizer` extracts raw keyword tokens from courses with noun phrase chunking.

class `cv_kickstarter.course_keyword_tokenizer.CourseKeywordTokenizer` (*course*)
 Bases: `object`

Can extract keyword tokens from course.

keyword_tokens ()

Return keyword tokens for course.

class `cv_kickstarter.course_keyword_tokenizer.CourseSentenceExtractor` (*course*)
 Bases: `object`

Can extract sentences from course object.

sentences ()

Return sentences for course.

The sentences are based on title, contents, `course_objectives_text` and `course_objectives`

class `cv_kickstarter.course_keyword_tokenizer.TextKeywordChunkifier`
 Bases: `object`

Can split text into several chunks by noun phrase chunking.

The chunks are extracted with noun phrase chunking with nouns and preceding adjectives (if there are any).

The code in this class is inspired by the NLTK book and Finn Nielsens code from Data Mining using Python 02819 from DTU.

chunks (*text*)

Return keyword chunks from the given text.

`cv_kickstarter.course_keyword_tokenizer.course_keyword_tokens` (*course*)

Return keyword tokens for course.

1.6 cv_kickstarter.course_repository module

Course Repository for storing and fetching courses from MongoDB.

class `cv_kickstarter.course_repository.Course`

Bases: tuple

`Course`(title, course_number, contents, course_objectives_text, course_objectives, tokens)

contents

Alias for field number 2

course_number

Alias for field number 1

course_objectives

Alias for field number 4

course_objectives_text

Alias for field number 3

title

Alias for field number 0

tokens

Alias for field number 5

class `cv_kickstarter.course_repository.CourseRepository` (*mongo_store*, *collection='courses'*)

Bases: object

Stores and fetches course objects.

CourseRepository is able to fetch course data and deserialize to course objects and store given course objects by serializing to JSON and store in MongoDB through the `mongo_store`.

create (*course*, *tokens*)

Create course with tokens in MongoDB by serializing into JSON.

find_by_course_number (*course_number*)

Find course by given course number in MongoDB.

remove (*course_number*)

Remove course with given `course_number` from MongoDB.

class `cv_kickstarter.course_repository.MongoStore` (*database_name*, *mongo_db_url=None*)

Bases: object

Database Client for MongoDB.

find (*collection*, *query*)

Find document in collection by query.

insert (*collection*, *hash_data*)

Insert `hash_data` (document) into collection.

remove (*collection, query*)
 Remove document in collection by query.

1.7 cv_kickstarter.cv_kickstarter_config module

Configuration for cv_kickstarter.

class cv_kickstarter.cv_kickstarter_config.**CvKickstarterConfig** (*config_file_path='app.cfg'*)
 Bases: object

Configuration for cv_kickstarter.

campus_net_app_name ()
 Return the campus net app name for CampusNet API.

Given by environment variable CAMPUS_NET_APP_NAME or in config file as [campusnet] app_name:
 my-app-name

campus_net_app_token ()
 Return the campus net app token name for CampusNet API.

Given by environment variable CAMPUS_NET_APP_TOKEN or in config file as [campusnet] app_token:
 secret-app-token

career_builder_key ()
 Return the career builder api key.

Given by environment variable CAREER_BUILDER_DEVELOPER_KEY or in config file as [campus-net] app_token: secret-app-token

go_key ()
 Return the go.dk api key.

Given by environment variable GO_DEVELOPER_KEY or in config file as [godk] guid: secret-guid-key

mongo_db_name ()
 Return the database name of the Mongo Database.

Given by environment variable MONGO_DB_NAME or in config file as [mongo] db_name:
 my_mongo_db_name

mongo_url ()
 Return the URL for the Mongo DB.

This is not mandatory to get the app to work.

Given by environment variable MONGO_URL.

secret_key ()
 Return the secret key for Flask app.

Given by environment variable SECRET_KEY or in config file as [flask] secret_key: my-secret-key

1.8 cv_kickstarter.dtu_course_base module

Structuring courses based on xml response from the DTU Course Base.

class cv_kickstarter.dtu_course_base.**Course**
 Bases: tuple

Course(title, course_number, contents, course_objectives_text, course_objectives)

contents

Alias for field number 2

course_number

Alias for field number 1

course_objectives

Alias for field number 4

course_objectives_text

Alias for field number 3

title

Alias for field number 0

class cv_kickstarter.dtu_course_base.**CourseExtractor** (course_xml, language)

Bases: object

Responsible for extracting course objects based on a course xml.

course ()

Return a Course object based on the given course xml.

cv_kickstarter.dtu_course_base.**courses_from_xml** (courses_xml_text, language='en-GB')

Extract course objects based on xml with courses.

1.9 cv_kickstarter.dtu_skill_set module

Extract the skill set of a DTU student.

This module is a layer on top of academic_skill_set that takes exam results as given by the CampusNet API, merges it with course information from the Course Base (through the course base repo).

class cv_kickstarter.dtu_skill_set.**CampusNetCourseBaseMerger** (exam_result_programmes,
course_base)

Bases: object

Merges exam results with information about the course.

course_exam_results ()

Return course exam results.

A list of ExamResult, that contains exam result information and more detailed information about the course given by the returned course object from course base.

class cv_kickstarter.dtu_skill_set.**DtuSkillSet** (exam_result_programmes,
course_base_repo)

Bases: object

Responsible for extracting a skill set based on DTU data sources.

skill_set ()

Return the skill set.

class cv_kickstarter.dtu_skill_set.**ExamResult** (grade, course, ects_points)

Bases: object

Exam result with grade, course and ects points.

course_tokens

Return the tokens of the course.


```
class cv_kickstarter.dtu_skill_set.TokenizedCourseExamResult
    Bases: tuple
    TokenizedCourseExamResult(exam_result, tokens, course)

    course
        Alias for field number 2

    exam_result
        Alias for field number 0

    tokens
        Alias for field number 1
```

1.10 cv_kickstarter.ects_grade_calculator module

Calculates average grade based on grades weighted by ECTS-point.

```
class cv_kickstarter.ects_grade_calculator.GradeAverageCalculator(exam_results)
    Bases: object
    Class that is able to calculate average based on ECTS exam_results.

    average_grade()
        Return the average grade.

cv_kickstarter.ects_grade_calculator.average_grade(exam_results)
    Return the average grade from a list of exam_results.
```

1.11 cv_kickstarter.nltk_data_downloader module

Downloads nltk data relevant for the cv kickstarter project.

If any new module needs data from nltk, it should be added here.

```
cv_kickstarter.nltk_data_downloader.download()
    Download relevant nltk data for cv kickstarter.

    If there already exists a nltk_data folder in the root, it is not downloaded.

    If nltk data is added the nltk_data folder need to be removed for downloading the new data.
```

1.12 cv_kickstarter.session_authentication module

Utilizes the a session dictionary as authentication storage.

Integrates with the Flask session.

```
class cv_kickstarter.session_authentication.SessionAuthentication(session_dict)
    Bases: object
    Utilizes the a session dictionary as authentication storage.

    auth_token
        Return authentication token from session.
```

authenticate (*student_id*, *auth_token*)

Authenticate a student by setting *student_id* and *auth_token*.

is_authenticated ()

Return true if the user is authenticated in the session.

log_out ()

Log the user out by removing *student_id* and *auth_token*.

student_id

Return student id from session.

1.13 Module contents

CVKickstarer - generate CV with skills and job recommendations.

CVKickstarter is a web app that is able to analyse your DTU information and provide you with a prefilled CV with your education information, courses you've passed, the performance of your study, the skills you've gained and job suggestions based on your best skills.

JOB_SEARCHER PACKAGE

2.1 Submodules

2.2 job_searcher.career_builder module

Module for job searching on CareerBuilder.com.

```
class job_searcher.career_builder.CareerBuilder (developer_key)  
    Bases: job_searcher.job_searcher.JobSearcher
```

JobSearcher for CareerBuilder.com.

```
BASE_URL = 'http://api.careerbuilder.com/v2/jobsearch'
```

```
PARAM_DEV_KEY = 'DeveloperKey'
```

```
PARAM_KEYWORDS = 'keywords'
```

```
PARAM_PER_PAGE = 'perpage'
```

```
find_results (keywords=[], amount=5)  
    Perform a job search.
```

Parameters **keywords** – Keywords, that should be contained in the returned results. :param **amount**: The amount of results wanted. :return: The jobs found by the given search parameters.

```
find_results_amount (keyword='')  
    Find the amount of results for a given keyword.
```

```
static xml_to_jobs (xml)  
    Convert xml to a list of jobs.
```

Parameters **xml** – The xml string, that should be parsed.

Returns A list of jobs.

2.3 job_searcher.go_jobs module

Module for job searching on Go.dk.

```
class job_searcher.go_jobs.GoJobs (guid)  
    Bases: job_searcher.job_searcher.JobSearcher
```

JobSearcher for Go.dk.

```
BASE_URL = u'http://moveon.dk/webservice/mobile.asmx/'
```

```
HEADERS = {u'Content-type': u'application/json', u'Accept': u'text/plain'}
```

```
PASS = u'02e19abe-b6f4-4a7e-bb70-9e613fcb43c2'
```

```
URL_EXTENSION_GET_JOB = u'GetJobLimitedV3'
```

```
URL_EXTENSION_SEARCH = u'SearchJobsV3'
```

```
find_results (keywords=(), amount=5)
```

Perform a job search.

Parameters **keywords** – Keywords, that should be contained in the returned

results. :param amount: The amount of results wanted. :return: The jobs found by the given search parameters.

```
find_results_amount (keyword=u'')
```

Find the amount of results for a given keyword.

```
get_details_for_job (job_id)
```

Get job details for the given job id.

Parameters **job_id** – Id for the job, that you want to retrieve.

Returns A job.

```
get_details_for_jobs (job_ids)
```

Get job details for the given job id's.

Parameters **job_ids** – Id's for the jobs, that you want to retrieve.

Returns A list of the given jobs.

```
static json_to_job (json_text)
```

Convert a json string to a job.

Parameters **json_text** – The json string, that should be parsed.

Returns A job.

2.4 job_searcher.job module

Generic job module.

```
class job_searcher.job.Job (title, company_name, teaser, job_url)
```

Generic Job interface.

Simple job class, used to expose a generic interface for jobs across different job searchers.

2.5 job_searcher.keyword_evaluator module

A module used for keyword evaluation.

```
class job_searcher.keyword_evaluator.KeywordEvaluator (job_searcher)
```

A class used to evaluate keywords with a given job searcher.

```
evaluate_keyword (keyword)
```

Evaluate a keyword for the given job searcher.

Parameters **keyword** – The keyword to evaluate.

Returns The percentage of jobs, that contain the given keyword.

2.6 Module contents

Job searching module for searching jobs and evaluate skill relevance.

USING CV KICKSTARTER

Data Mining using Python project repository

3.1 Setup

To install required modules, simply type:

```
python setup.py install
```

For optimization reasons, the course information is fetched from a MongoDB database. In order to import courses into MongoDB from the xml, run the command:

```
python course_import.py
```

3.2 Configuration

To configure the application, it is possible to either add an `app.cfg` file that contains the relevant configurations (see `app.cfg.example` for an example configuration file) or environment variables (see `cv_kickstarter_config.py` for environment variables used).

The app defaults to using MongoDB through localhost unless a `MONGO_URL` environment variable is given.

3.3 Run webserver

To run the web server after the setup, execute:

```
python webapp.py
```

or to run with gunicorn, execute:

```
gunicorn webapp:app --log-file=-
```

3.4 Command Line Integration

The CV can also be exported in a json format through the CLI by using the command:

```
jsoncv s123456 secret
```

where ‘s123456’ should be your student id and ‘secret’ be your password to CampusNet.

3.5 Tests

The project uses `py.test` for testing, so to run tests, execute:

```
py.test
```

The project is also set up with `tox` and is tested against python 2.7, 3.4, pypy and pypy3. To run tox locally, type:

```
python setup.py test
```


INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

C

`cv_kickstarter`, 14
`cv_kickstarter.academic_skill_set`, 4
`cv_kickstarter.cnapi`, 6
`cv_kickstarter.course_keyword_tokenizer`,
9
`cv_kickstarter.course_repository`, 10
`cv_kickstarter.cv_kickstarter_config`,
11
`cv_kickstarter.dtu_course_base`, 11
`cv_kickstarter.dtu_skill_set`, 12
`cv_kickstarter.ects_grade_calculator`,
13
`cv_kickstarter.models`, 4
`cv_kickstarter.models.exam_result_programme`,
3
`cv_kickstarter.models.user_cv`, 3
`cv_kickstarter.models.user_cv_builder`,
4
`cv_kickstarter.models.user_cv_dictionary_mapper`,
4
`cv_kickstarter.nltk_data_downloader`, 13
`cv_kickstarter.session_authentication`,
13

j

`job_searcher`, 17
`job_searcher.career_builder`, 15
`job_searcher.go_jobs`, 15
`job_searcher.job`, 16
`job_searcher.keyword_evaluator`, 16

A

AbstractXmlInfoExtractor (class in cv_kickstarter.cnapi), 7

auth_token (cv_kickstarter.session_authentication.SessionAuthentication attribute), 13

auth_token() (cv_kickstarter.cnapi.Authenticator method), 7

authenticate() (cv_kickstarter.cnapi.CampusNetApi method), 7

authenticate() (cv_kickstarter.session_authentication.SessionAuthentication method), 13

authenticate_with_token() (cv_kickstarter.cnapi.CampusNetApi method), 7

Authenticator (class in cv_kickstarter.cnapi), 7

average_grade (cv_kickstarter.models.exam_result_programme.ExamResultProgramme attribute), 3

average_grade() (cv_kickstarter.ects_grade_calculator.GradeAverageCalculator method), 13

average_grade() (in module cv_kickstarter.ects_grade_calculator), 13

B

BASE_URL (job_searcher.career_builder.CareerBuilder attribute), 15

BASE_URL (job_searcher.go_jobs.GoJobs attribute), 15

boosted_keyword_scores() (cv_kickstarter.academic_skill_set.KeywordGradeBooster method), 5

build() (cv_kickstarter.models.user_cv_builder.UserCVBuilder method), 4

C

campus_net_app_name() (cv_kickstarter.cv_kickstarter_config.CvKickstarterConfig method), 11

campus_net_app_token() (cv_kickstarter.cv_kickstarter_config.CvKickstarterConfig method), 11

CampusNetApi (class in cv_kickstarter.cnapi), 7

CampusNetCourseBaseMerger (class in cv_kickstarter.dtu_skill_set), 12

CampusNetExamResultMapper (class in cv_kickstarter.models.user_cv_builder), 4

career_builder_key() (cv_kickstarter.cv_kickstarter_config.CvKickstarterConfig method), 11

CareerBuilder (class in job_searcher.career_builder), 15

chunks() (cv_kickstarter.course_keyword_tokenizer.TextKeywordChunkifier method), 9

contents (cv_kickstarter.course_repository.Course attribute), 10

contents (cv_kickstarter.dtu_course_base.Course attribute), 12

Course (class in cv_kickstarter.course_repository), 10

Course (class in cv_kickstarter.dtu_course_base), 11

course (cv_kickstarter.dtu_skill_set.TokenizedCourseExamResult attribute), 13

course() (cv_kickstarter.dtu_course_base.CourseExtractor method), 12

course_exam_results() (cv_kickstarter.dtu_skill_set.CampusNetCourseBase method), 12

course_keyword_tokens() (in module cv_kickstarter.course_keyword_tokenizer), 10

course_number (cv_kickstarter.cnapi.ExamResult attribute), 8

course_number (cv_kickstarter.course_repository.Course attribute), 10

course_number (cv_kickstarter.dtu_course_base.Course attribute), 12

course_numbers (cv_kickstarter.academic_skill_set.CourseKeyword attribute), 5

course_objectives (cv_kickstarter.course_repository.Course attribute), 10

course_objectives (cv_kickstarter.dtu_course_base.Course attribute), 12

course_objectives_text (cv_kickstarter.course_repository.Course attribute), 10

course_objectives_text (cv_kickstarter.dtu_course_base.Course attribute), 12

course_title (cv_kickstarter.cnapi.ExamResult attribute), 8

course_title_sentence() (cv_kickstarter.models.user_cv.UserCV method), 3

course_tokens (cv_kickstarter.dtu_skill_set.ExamResult attribute), 12	exam_result (cv_kickstarter.dtu_skill_set.TokenizedCourseExamResult attribute), 13
CourseExtractor (class in cv_kickstarter.dtu_course_base), 12	exam_result() (cv_kickstarter.cnapi.ExamResultXmlMapper method), 8
CourseKeyword (class in cv_kickstarter.academic_skill_set), 4	exam_results (cv_kickstarter.cnapi.ProgramExamResults attribute), 8
CourseKeywordTokenizer (class in cv_kickstarter.course_keyword_tokenizer), 9	ExamResult (class in cv_kickstarter.cnapi), 7
CourseRepository (class in cv_kickstarter.course_repository), 10	ExamResult (class in cv_kickstarter.dtu_skill_set), 12
courses_from_xml() (in cv_kickstarter.dtu_course_base), 12	ExamResultProgramme (class in cv_kickstarter.models.exam_result_programme), 3
CourseSentenceExtractor (class in cv_kickstarter.course_keyword_tokenizer), 9	ExamResultXmlMapper (class in cv_kickstarter.cnapi), 8
CourseSkillSet (class in cv_kickstarter.academic_skill_set), 5	extract() (cv_kickstarter.cnapi.AbstractXmlInfoExtractor method), 7
CourseSkillSetMerger (class in cv_kickstarter.academic_skill_set), 5	F
create() (cv_kickstarter.course_repository.CourseRepository method), 10	filtered_skill_set() (cv_kickstarter.academic_skill_set.StudentSkillSetNoise method), 6
cv_kickstarter (module), 14	find() (cv_kickstarter.course_repository.MongoStore method), 10
cv_kickstarter.academic_skill_set (module), 4	find_by_course_number() (cv_kickstarter.course_repository.CourseRepository method), 10
cv_kickstarter.cnapi (module), 6	find_results() (job_searcher.career_builder.CareerBuilder method), 15
cv_kickstarter.course_keyword_tokenizer (module), 9	find_results() (job_searcher.go_jobs.GoJobs method), 16
cv_kickstarter.course_repository (module), 10	find_results_amount() (job_searcher.career_builder.CareerBuilder method), 15
cv_kickstarter.cv_kickstarter_config (module), 11	find_results_amount() (job_searcher.go_jobs.GoJobs method), 16
cv_kickstarter.dtu_course_base (module), 11	first_name (cv_kickstarter.cnapi.Student attribute), 8
cv_kickstarter.dtu_skill_set (module), 12	full_name (cv_kickstarter.models.user_cv.UserCV attribute), 3
cv_kickstarter.ects_grade_calculator (module), 13	G
cv_kickstarter.models (module), 4	get() (cv_kickstarter.cnapi.UserClient method), 9
cv_kickstarter.models.exam_result_programme (module), 3	get_details_for_job() (job_searcher.go_jobs.GoJobs method), 16
cv_kickstarter.models.user_cv (module), 3	get_details_for_jobs() (job_searcher.go_jobs.GoJobs method), 16
cv_kickstarter.models.user_cv_builder (module), 4	go_key() (cv_kickstarter.cv_kickstarter_config.CvKickstarterConfig method), 11
cv_kickstarter.models.user_cv_dictionary_mapper (module), 4	GoJobs (class in job_searcher.go_jobs), 15
cv_kickstarter.nltk_data_downloader (module), 13	grade (cv_kickstarter.cnapi.ExamResult attribute), 8
cv_kickstarter.session_authentication (module), 13	GradeAverageCalculator (class in cv_kickstarter.ects_grade_calculator), 13
CvKickstarterConfig (class in cv_kickstarter.cv_kickstarter_config), 11	grades (cv_kickstarter.models.user_cv_builder.UserCVBuilder attribute), 4
D	grades() (cv_kickstarter.cnapi.CampusNetApi method), 7
download() (in cv_kickstarter.nltk_data_downloader), 13	H
DtuSkillSet (class in cv_kickstarter.dtu_skill_set), 12	HEADERS (job_searcher.go_jobs.GoJobs attribute), 16
E	
ects_points (cv_kickstarter.cnapi.ExamResult attribute), 8	
email (cv_kickstarter.cnapi.Student attribute), 8	
evaluate_keyword() (job_searcher.keyword_evaluator.KeywordEvaluator method), 16	

highest_ranked_keywords
(cv_kickstarter.models.user_cv.UserCV attribute), 3

I

insert() (cv_kickstarter.course_repository.MongoStore method), 10
is_active (cv_kickstarter.cnapi.ProgramExamResults attribute), 8
is_authenticated() (cv_kickstarter.cnapi.CampusNetApi method), 7
is_authenticated() (cv_kickstarter.session_authentication.SessionAuthentication method), 14
is_done (cv_kickstarter.models.exam_result_programme.ExamResultProgramme attribute), 3

J

Job (class in job_searcher.job), 16
job_searcher (module), 17
job_searcher.career_builder (module), 15
job_searcher.go_jobs (module), 15
job_searcher.job (module), 16
job_searcher.keyword_evaluator (module), 16
json_to_job() (job_searcher.go_jobs.GoJobs static method), 16

K

keyword (cv_kickstarter.academic_skill_set.CourseKeyword attribute), 5
keyword_scorer() (cv_kickstarter.academic_skill_set.KeywordScoreCalculator method), 5
keyword_tokens() (cv_kickstarter.course_keyword_tokenizer.CourseKeywordTokenizer method), 9
KeywordEvaluator (class in job_searcher.keyword_evaluator), 16
KeywordGradeBooster (class in cv_kickstarter.academic_skill_set), 5
KeywordScoreCalculator (class in cv_kickstarter.academic_skill_set), 5
KeywordScoreNormalizer (class in cv_kickstarter.academic_skill_set), 5

L

last_name (cv_kickstarter.cnapi.Student attribute), 8
log_out() (cv_kickstarter.session_authentication.SessionAuthentication method), 14

M

mapped_exam_result() (cv_kickstarter.models.user_cv_builder.CampusNetExamResultMapper method), 4
mongo_db_name() (cv_kickstarter.cv_kickstarter_config.CvKickstarterConfig method), 11
mongo_url() (cv_kickstarter.cv_kickstarter_config.CvKickstarterConfig method), 11

MongoStore (class in cv_kickstarter.course_repository), 10

N

name (cv_kickstarter.cnapi.ProgramExamResults attribute), 8
normalized_keyword_score() (cv_kickstarter.academic_skill_set.KeywordScoreNormalizer method), 5

P

PARAM_AUTHORIZATION_KEY (job_searcher.career_builder.CareerBuilder attribute), 15
PARAM_KEYWORDS (job_searcher.career_builder.CareerBuilder attribute), 15
PARAM_PER_PAGE (job_searcher.career_builder.CareerBuilder attribute), 15
PASS (job_searcher.go_jobs.GoJobs attribute), 16
passed_ects_points (cv_kickstarter.cnapi.ProgramExamResults attribute), 8
period (cv_kickstarter.cnapi.ExamResult attribute), 8
ProgramExamResults (class in cv_kickstarter.cnapi), 8

R

rank (cv_kickstarter.academic_skill_set.CourseKeyword attribute), 5
remove() (cv_kickstarter.course_repository.CourseRepository method), 10
remove() (cv_kickstarter.course_repository.MongoStore method), 10

S

secret_key() (cv_kickstarter.cv_kickstarter_config.CvKickstarterConfig method), 11
sentences() (cv_kickstarter.course_keyword_tokenizer.CourseSentenceExtractor method), 9
SessionAuthentication (class in cv_kickstarter.session_authentication), 13
skill_set() (cv_kickstarter.academic_skill_set.CourseSkillSet method), 5
skill_set() (cv_kickstarter.academic_skill_set.StudentSkillSet method), 6
skill_set() (cv_kickstarter.dtu_skill_set.DtuSkillSet method), 12
skill_set() (in module cv_kickstarter.academic_skill_set), 6
Student (class in cv_kickstarter.cnapi), 8
student_id (cv_kickstarter.session_authentication.SessionAuthentication attribute), 14
student_skill_set() (cv_kickstarter.academic_skill_set.CourseSkillSetMerge method), 5
StudentSkillSet (class in cv_kickstarter.academic_skill_set), 5

StudentSkillSetNoiceFilter (class in
cv_kickstarter.academic_skill_set), 6

T

TextKeywordChunkifier (class in
cv_kickstarter.course_keyword_tokenizer),
9

title (cv_kickstarter.course_repository.Course attribute),
10

title (cv_kickstarter.dtu_course_base.Course attribute), 12

TokenizedCourseExamResult (class in
cv_kickstarter.dtu_skill_set), 12

tokens (cv_kickstarter.course_repository.Course at-
tribute), 10

tokens (cv_kickstarter.dtu_skill_set.TokenizedCourseExamResult
attribute), 13

U

URL_EXTENSION_GET_JOB
(job_searcher.go_jobs.GoJobs attribute),
16

URL_EXTENSION_SEARCH
(job_searcher.go_jobs.GoJobs attribute),
16

user (cv_kickstarter.models.user_cv_builder.UserCVBuilder
attribute), 4

user() (cv_kickstarter.cnapi.CampusNetApi method), 7

user_cv_dict() (cv_kickstarter.models.user_cv_dictionary_mapper.UserCVDictionaryMapper
method), 4

user_id (cv_kickstarter.cnapi.Student attribute), 8

user_picture() (cv_kickstarter.cnapi.CampusNetApi
method), 7

UserClient (class in cv_kickstarter.cnapi), 8

UserCV (class in cv_kickstarter.models.user_cv), 3

UserCVBuilder (class in
cv_kickstarter.models.user_cv_builder), 4

UserCVDictionaryMapper (class in
cv_kickstarter.models.user_cv_dictionary_mapper),
4

UserGradesExtractor (class in cv_kickstarter.cnapi), 9

UserInfoExtractor (class in cv_kickstarter.cnapi), 9

W

word_scorer() (cv_kickstarter.academic_skill_set.WordFrequencyScoreCalculator
method), 6

WordFrequencyScoreCalculator (class in
cv_kickstarter.academic_skill_set), 6

X

xml_to_jobs() (job_searcher.career_builder.CareerBuilder
static method), 15

Y

year (cv_kickstarter.cnapi.ExamResult attribute), 8