Hamsa Survey's doc

version

Bruno Ferraz

August 31, 2020

Contents

Welcome to Hamsa Survey’s documentation!

hamsa

hamsa package

Submodules

hamsa.heuristics module

This module is used to store all heuristics values as constants.

constant int THRESHOLD_UNIQUE: Threshold used to identify opened questions. If you have more unique answers than the THRESHOLD_UNIQUE probably you have a opened question. The threshold is expressed in percent.

`hamsa.heuristics.`**`THRESHOLD_UNIQUE`** = *50*

Constant int THRESHOLD_UNIQUE_MULTIPLE_CHOICE: Threshold used to identify multiple choice questions from closed ones

hamsa.instance module

class `hamsa.instance.`**`IInstance`**

Bases: **`object`**

Interface to define classes to hold and navigate through reference to surveys entries

abstract **`get_answer`** **(**columnIndex**)**

abstract **`get_answers`** **()**

abstract **`get_question`** **(**columnIndex**)**

abstract **`get_question_heading`** **(**columnIndex**)**

class `hamsa.instance.`**`Instance`** (surveyParam, rowIndex: int)

Bases: **`hamsa.instance.IInstance`**

`get_answer` **()**

`get_answers` **()**

`get_question` **()**

`get_question_heading` **()**

hamsa.question module

class `hamsa.question.`**`IQuestionType`**

Bases: **`object`**

Interface to define the methods used to decode answers from string base. The behavior used to this processed will change due to the question type. This abstraction works like a Type for the question.

abstract **`pre_process`** **()**

abstract **get_answers ()**

abstract **get_answers_encoded ()**

abstract **get_categories ()**

class hamsa.question.**QuestionType** (value)

Bases: **enum.Enum**

An enumeration.

UNKNOW = *0*

OPENED = *1*

CLOSED = *2*

CLOSED_MULTIPLE_CHOICE = *3*

CLOSED_CHECKBOX = *4*

class hamsa.question.**IQuestion**

Bases: **object**

Public Methods

abstract **get_id ()** → int

abstract **get_label ()**

abstract **get_heading (**i: int**)**

abstract **get_type ()**

abstract **get_categories ()**

abstract **get_type_string ()**

abstract **get_raw_answers (**i: int**)**

abstract **get_answers_encoded ()**

abstract **get_state ()**

abstract **set_state (**state: bool**)**

abstract **change_type ()**

class hamsa.question.**ConcreteQuestionsType** (q)

Bases: **hamsa.question.IQuestionType**

pre_process ()

get_answers ()

get_categories ()

get_answers_encoded ()

class hamsa.question.**OpenEndedType** (q)

Bases: **hamsa.question.ConcreteQuestionsType**

get_answers ()

class hamsa.question.**ClosedEndedType** (q)

Bases: **hamsa.question.ConcreteQuestionsType**

get_answers ()

class hamsa.question.**ClosedEndedMultipleChoiceType** (q)

Bases: **hamsa.question.ClosedEndedType**

pre_process ()

get_answers ()

get_answers_encoded ()

get_categories ()

class hamsa.question.**Question** (surveyParam, columnIndex: int, questionLabel: str)

Bases: **hamsa.question.IQuestion**

get_id ()

get_label ()

get_heading ()

get_type ()

Method to get the question type

Return QuestionType: Enum that indicate the type. Must be decoded

get_type_string ()

Method to get the question type already decoded into string

Return str: QuestionType enum decoded

get_raw_answers ()

get_answers_encoded ()

get_categories ()

get_state ()

Get the question state If True, it will be exported It False, it will not be exported

set_state (state: bool**)**

Set the question state. If it join the export or not

change_type (questionTypeParam: hamsa.question.QuestionType**)**

Method used to change question type. It already invoke the the private method _create_type. There, the state will be created since it have not been created already

Parameters: **questionTypeParam** (*QuestionType*) – Type of question you want to change into.

hamsa.survey module

class hamsa.survey.**ISurvey**

Bases: **object**

Interface works as a FAÇADE for hamsa survey system. Futhermore, also works as MODEL from MVC pattern *abstract* **`pre_process` ()**

abstract **`get_question` (**i: int**)**

abstract **`get_question_heading` (**i: int**)**

abstract **`get_questions` ()** → list

abstract **`get_questions_headings` ()** → list

abstract **`get_questions_labels` ()** → list

abstract **`get_questions_states` ()** → list

abstract **`get_questions_categories` ()** → list

`get_questions_types` () → list

abstract **`get_questions_by_type` ()** → list

abstract **`get_questions_by_state` ()** → list

abstract **`get_instance` (**i: int**)**

abstract **`get_answer` (**i: int**,** j: int**)**

abstract **`get_question_raw_answers` (**columnParam: int**)**

abstract **`get_instance_raw_answers` (**rowindex: int**)**

abstract **`get_report` ()**

abstract **`get_report_data` ()**

abstract **`export_to_matlab` (**path**)**

class `hamsa.survey.`**`Survey`** (df)
Bases: **`hamsa.survey.ISurvey`**
Data Structure that stores information about how data will be exported to neural network

`pre_process` ()
Pre-process the data provided figuring out the questions type

`get_question` (index**)**
Get a Question object from the pre-processed list

Parameters: **index** – Question's index position

Return Question: Question object referenced by index

`get_question_heading` (index**)**
Get specific question's heading referenced by the index

Parameters: **index** (*int*) – Index position that

`get_questions` () → list
Get a list of pre-processed question

Return list: Questions pre-processed

`get_questions_headings` () → list

Get a list with question's headings

Return list: Question's Headings

`get_questions_states` **()** → list

`get_questions_categories` **()** → list

Get a list of lists from the question's categories. If the question is opened the request will be None

`get_questions_by_type` **(**typewanted: hamsa.question.QuestionType**)** → list

Get a list of questions filtered by type

Parameters: **typewanted** (*QuestionType*) –

Return list: Questions filtered by given type

`get_questions_by_state` **(**statewanted: bool**)** → list

Get a list of questions filtered by state

Parameters: **statewanted** (*bool*) – True means enabled False means disable

Return list: Questions filtered by given state

`get_questions_labels` **()** → list

`get_questions_types` **()** → list

`get_instance` **(**rowindex**)**

Get a Instance object from the list

Return Instance:

`get_answer` **(**lineParam**,** columnParam**)**

Get an specific an answer from a specific question

Parameters:

- **lineParam** (*int*) – line requested
- **columnParam** (*int*) – column requested

Return str: Answer requested

`get_question_raw_answers` **(**columnParam**)** → pandas.core.series.Series

Get answers from a specific question

Parameters: **columnParam** (*int*) – column requested

Return pandas.Series: answers organized in a pandas.Series

`get_instance_raw_answers` **(**rowindex**)**

Get answers from a specific instance

Parameters: **rowindex** (*int*) – index requested

Return list: answers organizes in a list

`get_report` **()**

Generate statistics based on the data provided and return it as a string

Return str: report string

`get_report_data` **()**

Return data to be used in Report screen

`export_to_matlab` **(**path**)**

Module contents

`hamsa.read_csv` (path=None, token=',', encoding='utf8') → hamsa.survey.Survey

Reads a CSV file

Parameters:

- **path** (*str*) – indicates the file and the path to it
- **token** (*str*) – token used to split the a file along the parsing step. The default is "," but ";" or "/t" are largely used instead. Opened questions demands diferent tokens since the interviewed could write anything.
- **encoding** (*str*) – assign which encoding must be used along the parsing step. The default is UTF8

Return survey.Survey: The object Survey

Indices and tables

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

Index

Python Module Index