PyTorch-MT

Generated by Doxygen 1.8.11

Contents

Chapter 1

Configs

The application has to be provided with two main components, a model, that will be used to solve a given a problem, and task, which describes the problem, and the way that the model will be used to solve it.

According to the previously defined structure, the application currently accepts the model and task descriptors in the form of **JSON** files.

There are several types of descriptors:

1. Model
2. Experiment
3. Language
4. Policy
5. Translator
6. Reguralizer

The first step to start an experiment, is to create it's configuration file. The specific parameters, which must be provided in the configuration are described in the class-level attribute `interface` of the experiment. All classes, which take part in the configuration assembly mechanism, must inherit from `Component` abstract base class. This provides the required class level fields, which are the previously mentioned `interface` (`Interface` type object) and `abstract` (bool value). During the assembly mechanism the builder object parses the provided configuration file, and detects the type of the node, which is given in the `type` entry of the **JSON**. It then iterates through the `interface` attribute of the detected class, and fetches the required parameters from the `params` entry of the configuration file. During this step, the builder may find an entry, that can't be given as a single parameter, since it may also be a complex component type object, such as the currently assembled experiment. There are multiple ways to define these objects, but the most common is to create a **JSON** object, that contains the type and its parameters. If the list of parameters is too long, or this object also contains a complex type, then it can also be a symbolic link to a file, that contains the configuration. Another way to create a complex parameter, is to define multiple instances in the configuration file, by either packing them in a list, or a **JSON** object. If the currently assembled object does not require any identifier paired with the complex instances, then it is enough to pass it as a list, otherwise the builder object will create a python dictionary, containing the entries with the defined name.

Experiment configurations

Experiment configurations must follow the format, which is described in the following examples.

```
{
    "type": "<Experiment Type>",
    "params": {

    },
    "model_dir": "<Output Path>"
}
```

The previous **JSON** file is a general scheme for an experiment configuration file. The value of `type` entry defines the experiment, and the `params` entry contains a **JSON** object for declaring the parameters. `model_dir` contains the location of the outputs and logs for the experiment.

The following configuration, with the corresponding interface definition may be a viable description for an experiment.

```
interface = Interface(**{
        'policy':               (0, Policy),
        'language_identifiers': (1, None),
        'languages':            (2, Language),
        'model':                (3, Model),
        'initial_translator':   (4, WordTranslator),
        'reguralizer':          (5, Classifier)
})
```

```
{
    "type": "MergedCurriculumTranslation",
    "params": {
        "policy": "configs/utils/policies/policy.json",
        "language_identifiers": [
            "<ENG>",
            "<FRA>"
        ],
        "languages": [
            "configs/utils/languages/english.json",
            "configs/utils/languages/french.json"
        ],
        "model": "configs/models/sts.json",
        "initial_translator": "configs/utils/translators/word.json",
        "reguralizer": "configs/components/reguralizers/mlp.json"
    },
    "model_dir": "model_outputs/unmt_3"
}
```

The type of the experiment is `MergedCurriculumTranslation`, that requires a `policy`, which is a complex parameter, that defines specific behaviours for the model during training, validation or testing phase. `language_identifiers` is a primitive, that should contain a list of strings, which will identify the languages, that are used in the experiment. `languages` is a complex parameter, that yields its values as a list. A `Language` object defines the input pipelines and the vocabulary for a language. `model` is the configuration for the model, that will be in the experiment. For further examples see the pre-defined experiment configurations in the ∗/configs/tasks∗ directory.

Model configurations

Model configurations must follow the format, which is described in the following examples.

```
{
    "type": "<Model Type>",
    "params": "<Model Components"

}
```

The following example shows a possible configuration for the model.

```
{
    "type": "SeqToSeq",
    "params": {
        "encoder":  {
            "type": "UnidirectionalRNNEncoder",
            "params": {
                "hidden_size": 100,
                "recurrent_type": "LSTM",
                "num_layers": 3,
                "optimizer_type": "Adam",
                "learning_rate": 0.01
            }
        },
        "decoder": {
            "type": "RNNDecoder",
            "params": {
                "hidden_size": 100,
                "recurrent_type": "LSTM",
                "num_layers": 3,
                "optimizer_type": "Adam",
                "learning_rate": 0.01,
                "max_length": 15
            }
        }
    }
}
```

The type for the model was defined as sequence to sequence, which requires an encoder and decoder as it's components. *(To learn more about the required parameters for a given model, see the corresponding README.md files in the modules directories.)* In case of the `SeqToSeq` model, `encoder` and `encoder` are required nodes, and there will be an error message indicating their absence. After parsing the defined type of components, the application will look for the parameters required for the instantiation. The parameters shown in the example are *(currently)* sufficient for any of the `encoder` or `encoder` type modules.

Chapter 2

nmt-BMEVIAUAL01

Implementation of an unsupervised neural machine translation algorithm in PyTorch.

The algorithm is based on https://arxiv.org/abs/1711.00043.

Chapter 3

Components

Components are the main building blocks of the models. In particular, the sequence-to-sequence type models are well-suited for these modular elements. In the current state of the API, there are two distinct versions of the components, encoders and decoders. Each of these have 3 different methods:

1. Recurrent
2. Convolutional
3. Quasi-Recurrent

Encoders

Recurrent

1. Unidirectional encoder
2. Bidirectional encoder

Unidirectional type encoders could be considered as the regular recurrent units, which may yield different methods for calculations. The currently implemented features are the LSTM-type units and GRUs. The problem with these type of architectures, is the ability to preserve references in long sequences. Even the LSTMs and GRUs can't seem to resolve dependencies in longer, 40-50 unit length sequences. As a solution Bidirectional encoders start their operations from the end of the sentence, going 'backward' in time, as well as from the start of the sequence. This way there are 2 hidden states, one for each direction, which will then be concatenated, and fed to the upcoming layer. This method shortens the path between dependencies, and performs considerably better in numerous tasks.

Convolutional

COMING SOON

Quasi-Recurrent

COMING SOON

Decoders

Recurrent

1. Regular decoder
2. Attentional decoder
   (a) Bahdanau-style
   (b) Luong-style
      i. Dot Attention Decoder
      ii. General Attention Decoder
      iii. Concat Attention Decoder

Regular

Similarly to the encoders, the basic recurrent decoders also operate with an LSTM or GRU. The encoder provides the starting hidden state for the component, that contains the encoded latent representation of the source language. The decoder then starts to unfold this hidden state by predicting the first word of the target sentence, which will be then fed to the decoder at the next time step. This phase goes until the decoder predicts an <EOS> token. Although this simple method is the fastest, there are other techniques, which provide much better performance.

Bahdanau-style Attention Decoder

Considering the method of translation from a human viewpoint, the encoder-decoder method of machine translation may not be a very intuitive approach, since when the decoder tries to predict the most probable word at the first position, it takes the whole encoded sentence into account. It would be much more natural, if the decoder would only consider those parts of the source sentence, which highly correlate to the currently decoded word of the target sentence. `Neural Machine Translation by Jointly Learning to Align and Translate` introduces a method for this approach, that is called attention, which is an existing techniques in image related machine learning tasks, but has not been applied to natural language processing yet.

The main idea is to integrate another layer between the decoder and encoder, which will operate this mechanism. At each decoding step this layer calculates a weight distribution over the outputs of the encoder at each encoding time step. The new encoded latent state will come from the linear combination of the encoder hidden states, with their corresponding weights.

Although the core concept of attention is the same, there are different methods for calculating the weights for the encoder outputs. The already mentioned Bahdanau-style method uses a trainable layer, which takes the concatenation of the investigated encoder state and the decoder state the at the previous time step. The output of this operation is a single scalar value (or a vector of values in case of batched calculations), that will be the weight for the used encoder state. After each of the encoder output states have been weighted, the recurrent layer receives the weighted sum of these vectors.

Luong-style Attention Decoder

Another approach has been introduced by `Effective Approaches to Attention-based Neural Machine Translation`, which alters the order of weight calculation, and defines several alternative techniques for the scoring mechanism. Compared to the Bahdanau method `recurrent(score(h_t-1), i_t)` where i_t is the output of the previous decoding time step, and h_t-1 is the hidden state of the previous time step, the Luong method calculates `score(recurrent(h_t-1, i_t))`. The scoring of hidden state with respect to the encoder outputs, happens at the t-th time step, which will be used by a final output layer, that projects the hidden state to the vocabulary, with a softmax activation.

As mentioned above, Luong introduced 2 new methods for weight calculation (score function) additional to the method used in Bahdanau's work.

1. *Dot Attention Decoder*

The simplest and fastest scoring function, which according to my experience, helps the convergence of the model, better than any other scoring methods. The weight simply comes from the dot product of the encoder state and the hidden state.

```
h_d * h_eT
```

1. *General Attention Decoder*

This method uses a trainable layer similar to the concatenative methods, but instead of concatenation, it takes the dot product of the weight layer, with the encoder output state, and then the dot product with the decoder hidden state.

```
h_d * (W_a * h_eT)
```

The positive impact of this method compared to the simple dot product, is the constrain of creating a good attention weight distribution over the encoder output states is the responsibility of the attention weight layer, instead of the encoder and decoder recurrent layers.

1. *Concat Attention Decoder*

This method is the same as the one used in Bahdanau's experiments, which takes the concatenation of the decoder hidden state and encoder output state, with a dedicated attention weight layer.

```
v_t * tanh(W_a * [h_d ; h_e])
```

Convolutional

COMING SOON

Quasi-Recurrent

COMING SOON

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 7

Namespace Documentation

7.1 add_tokens Namespace Reference

Functions

- def **main** ()

Variables

- string **DEFAULT_FILE_INPUT** = '/media/patrik/1EDB65B8599DD93E/data/server/fra/fra_data'
- string **DEFAULT_FILE_OUTPUT** = '/media/patrik/1EDB65B8599DD93E/data/server/fra/fra_data_seg'

7.1.1 Detailed Description

7.2 align_vocabs Namespace Reference

Functions

- def **measure_length** (path)
- def **load_vocab** (path, desc, shape)
- def **main** ()

Variables

- string **DEFAULT_VOCAB_SRC** = '/media/patrik/1EDB65B8599DD93E/data/mapping/vectors-en-sync.txt'
- string **DEFAULT_VOCAB_TGT** = '/media/patrik/1EDB65B8599DD93E/data/mapping/vectors-fr-sync.txt'
- string **DEFAULT_ALIGNMENT_PATH_SRC** = '/media/patrik/1EDB65B8599DD93E/data/eng/ALIGNMEN↩T_EN_to_FR_2'
- string **DEFAULT_ALIGNMENT_PATH_TGT** = '/media/patrik/1EDB65B8599DD93E/data/fra/ALIGNMENT↩_FR_to_EN_2'
- string **DEFAULT_MAPPING_PATH** = '/media/patrik/1EDB65B8599DD93E/data/mapping/best_mapping.pth'
- int **DEFAULT_DIM** = 300
- int **DEFAULT_SIZE_SRC** = 0
- int **DEFAULT_SIZE_TGT** = 0

7.2.1 Detailed Description

7.3 analysis Namespace Reference

Classes

- class Analyzer
- class AttentionData
- class Data
- class DataLog
- class DataLogContainer
- class LatentStateData
- class Plot
- class ScalarData
- class TextData

Functions

- def create_report (corpora, vocab)
- def create_embedding_analyzer (vocab_paths, save_path, dimension=2, analyzer_type='PCA')
- def analyze_embeddings (vocab_paths, words, analyzer_path, dim=2)

7.3.1 Detailed Description

7.3.2 Function Documentation

7.3.2.1 def analysis.analyze_embeddings (*vocab_paths, words, analyzer_path, dim =* 2)

```
Arguments:
    vocab_paths:

    words:

    analyzer_path:

    dim:
```

7.3.2.2 def analysis.create_embedding_analyzer (*vocab_paths, save_path, dimension =* 2*, analyzer_type =* 'PCA')

```
Arguments:
    vocab_paths:

    dimension:

    save_path:

    analyzer_type:
```

7.3.2.3 def analysis.create_report (*corpora,* *vocab*)

```
Arguments:
    corpora:

    vocab:
```

7.4 config Namespace Reference

Classes

- class Config

7.4.1 Detailed Description

7.5 create_vocab Namespace Reference

Functions

- def **vocab_creator** (lang)
- def **main** ()

Variables

- int **EMBEDDING_DIM** = 50
- dictionary **ENG**
- dictionary **FRA**

7.5.1 Detailed Description

7.5.2 Variable Documentation

7.5.2.1 dictionary create_vocab.ENG

Initial value:

```
= {
    'vocab':    '/media/patrik/1EDB65B8599DD93E/data/server/eng/eng_vocab',
    'corpora':  '/media/patrik/1EDB65B8599DD93E/data/server/eng/ENG_DATA_SYNC'
}
```

7.5.2.2 dictionary create_vocab.FRA

Initial value:

```
= {
    'vocab':    '/media/patrik/1EDB65B8599DD93E/data/server/fra/FRA_VOCAB',
    'corpora':  '/media/patrik/1EDB65B8599DD93E/data/server/fra/FRA_DATA_SYNC'
}
```

7.6 divide_corpora Namespace Reference

Functions

- def **location_scheme** (path, corpora_type)
- def **main** ()

Variables

- string **DEFAULT_CORPORA** = '/media/patrik/1EDB65B8599DD93E/data/server/fra/fra_data_seg'
- float **DEFAULT_TRAIN_SPLIT** = 0.975
- float **DEFAULT_DEV_SPLIT** = 0.0125

7.6.1 Detailed Description

7.7 experiments Namespace Reference

Classes

- class DividedCurriculumTranslation
- class Experiment
- class MergedCurriculumTranslation
- class UnsupervisedTranslation

Functions

- def **__init__**
- def **train**
- def validate (self)
- def **test** (self)
- def **evaluate** (self)
- def state (self)
- def state (self, state)

Variables

- interface = UnsupervisedTranslation.interface
- bool **abstract** = False
- **reguralize**

7.7.1 Detailed Description

```
@package experiments
```

7.7.2 Function Documentation

7.7.2.1 def experiments.state (*self*)

```
Property for the state of the task.
```

7.7.2.2 def experiments.state (*self*, *state*)

```
Setter function for the state of the task, and the embeddings.
```

7.7.2.3 def experiments.validate (*self*, *dict*)

```
This function evaluates the model. Input data is propagated forward, and then the loss calculated
based on the same loss function which was used during training. The weights however, are not modified
in this function.

:return logs:
    A list of DataLog type objects, that contain the logging data for the languages. The number of
    data logs equal to the number of languages, and each data log contains information about the
    produced output for the whole data set of a language.

        total_loss:
            The total loss of the iteration, which is the same as the model loss during training.
            The value contains the loss of translation, auto-encoding and reguralization loss. The
            individual error of the discriminator is not included.

        translation_loss:
            The error, that is produced by the model, when translating a sentence.

        auto_encoding_loss:
            The error, that is produced by the model,
            when restoring (auto-encoding) a sentence.

        reguralization_loss:
            The reguralization loss, that is produced by the discriminator.

        discriminator_loss:
            The error of the discriminator, which is the loss that is produced, when the
            discriminator identifies a given latent vector.

        translation_text:
            The textual representation of the input, target and output symbols at the
            translation phase. These texts are produced by the format outputs
            utility function.

        auto_encoding_text:
            The textual representation of the input, target and output symbols at the
            auto encoding phase. These texts are produced by the format outputs
            utility function.

        Additional outputs depend on the chosen model.
```

7.7.3 Variable Documentation

7.7.3.1 experiments.interface = UnsupervisedTranslation.interface

7.8 generate Namespace Reference

Functions

- def **location_scheme** (path, corpora_type)
- def **synchronize** (vocab_path, new_vocab_path, corpora_path, new_corpora_path, oov_limit, min_length)
- def **create_vocab** (vocab_path, corpora_path, embedding_dim)
- def **add_tokens** (corpora_path, new_corpora_path)
- def **split_corpora** (corpora_path, train_split, validation_split)
- def **synchronize_vocabs** (candidate_vocab_path, reference_vocab_path, new_vocab_path)
- def **measure_length** (path)
- def **load_vocab_for_alignment** (path, desc, shape)
- def **align_vocabs** (source_vocab_path, target_vocab_path, alignment_embedding_dimension)
- def **remove_unique** (corpora_path, new_corpora_path, min_count, max_removed, min_length, max_↩ vocab_size, max_corpora_size)
- def **tokenize** (word)
- def **tokenize_corpora** (corpora_path, new_corpora_path, min_length, max_length)
- def **remove_duplicates** (vocab_path, new_vocab_path, dimension)
- def **main** ()

7.8.1 Detailed Description

7.9 models Namespace Reference

Classes

- class Model
- class SeqToSeq

7.9.1 Detailed Description

7.10 modules Namespace Reference

Classes

- class _STSModule
- class AutoEncoder
- class Discriminator
- class NoiseModel
- class Translator
- class WordTranslator

7.10.1 Detailed Description

7.11 qrnn Namespace Reference

Classes

- class QRNNDecoder
- class QRNNEncoder

Functions

- def optimizers (self)
- def **state** (self)
- def **__init__** (self)
- def **forward** (self, inputs, lengths, hidden)
- def **embedding** (self)
- def **embedding** (self, embedding)
- def **hidden_size** (self)

7.11.1 Detailed Description

7.11.2 Function Documentation

7.11.2.1 def qrnn.optimizers (*self*)

```
Quasi-recurrent decoder module of the sequence to sequence model.
```

```
Quasi-recurrent encoder module of the sequence to sequence model.
```

7.12 reader Namespace Reference

Classes

- class Bilingual
- class Corpora
- class DataQueue
- class FileInput
- class InputPipeline
- class Language
- class MemoryInput
- class Monolingual
- class Padding
- class ParallelDataQueue
- class PostPadding
- class PrePadding
- class Vocabulary

Functions

- def **__init__**
- def initialize_corpus (self)
- def source_vocabulary (self)
- def target_vocabulary (self)
- def source_vocab_size (self)
- def target_vocab_size (self)
- def generator (self)

Variables

- interface
- bool **abstract** = False
- int MAX_SEGMENT = 50000

7.12.1 Detailed Description

7.12.2 Function Documentation

7.12.2.1 def reader.generator (*self*, *list*)

```
Data is retrieved directly from a file, and loaded into data chunks of size MAX_CHUNK_SIZE.
```

7.12.2.2 def reader.initialize_corpus (*self*)

7.12.2.3 def reader.source_vocab_size (*self*)

```
Property for the vocab size of the source language.
:return: int, number of words in the source language.
```

7.12.2.4 def reader.source_vocabulary (*self*)

```
Property for the source language of the text corpora.
:return: Language, instance of the wrapper class for the source language.
```

7.12.2.5 def reader.target_vocab_size (*self*)

```
Property for the vocab size of the target language.
:return: int, number of words in the target language.
```

7.12.2.6 def reader.target_vocabulary (*self*)

```
Property for the target language of the text corpora.
:return: Language, instance of the wrapper class for the target language.
```

7.12.3 Variable Documentation

7.12.3.1 reader.interface

Initial value:

```
= Interface(**{
        ’data_path’:          (0, None),
        ’separator’:          (1, None),
        ’vocabulary’:         (2, ’Vocabulary$’),
        ’cuda’:               (3, ’Experiment:Policy:cuda$’)
    })
```

```
Wrapper class for the corpora, that yields two languages. The languages are paired, and
are separated by a special separator token.
```

7.12.3.2 int reader.MAX_SEGMENT = 50000

```
A queue object for the data feed. This can be later configured to load the data to
memory asynchronously.
```

7.13 rnn Namespace Reference

Classes

- class AttentionRNNDecoder
- class BahdanauAttentionRNNDecoder
- class BidirectionalRNNEncoder
- class ConcatAttentionRNNDecoder
- class DotAttentionRNNDecoder
- class GeneralAttentionRNNDecoder
- class LuongAttentionRNNDecoder
- class RNNDecoder
- class RNNEncoder
- class UnidirectionalRNNEncoder

7.13.1 Detailed Description

7.14 session Namespace Reference

Classes

- class EvaluationContext
- class Session
- class TestContext
- class TrainingContext
- class ValidationContext

Functions

- def **average_as_list** (outputs, key)

7.14.1 Detailed Description

7.15 substitute Namespace Reference

Functions

- def **similarity_predicate_long** (distance, word_len)
- def **similarity_predicate_short** (distance, word_len)
- def **find_best_match** (string, vocab)
- def **main** ()

Variables

- **str**
- **int**

7.15.1 Detailed Description

7.16 synchronize Namespace Reference

Functions

- def **main** ()

Variables

- string **DEFAULT_CORPORA** = '/media/patrik/1EDB65B8599DD93E/data/eng/ENG_DATA'
- string **DEFAULT_VOCAB** = '/media/patrik/1EDB65B8599DD93E/data/mapping/vectors-en.txt'
- string **DEFAULT_OUTPUT_CORPORA** = '/media/patrik/1EDB65B8599DD93E/data/mapping/engdata'
- string **DEFAULT_OUTPUT_VOCAB** = '/media/patrik/1EDB65B8599DD93E/data/mapping/vectors-en-sync.txt'
- int **DEFAULT_OOV_LIMIT** = 3
- int **DEFAULT_MIN_LENGTH** = 3

7.16.1 Detailed Description

7.17 tokenize_corpora Namespace Reference

Functions

- def **tokenize** (word)
- def **main** ()

Variables

- string **DEFAULT_INPUT** = '/media/patrik/1EDB65B8599DD93E/data/eng/test'
- string **DEFAULT_OUTPUT** = '/media/patrik/1EDB65B8599DD93E/data/eng/test_tok'
- int **DEFAULT_MIN** = 3
- int **DEFAULT_MAX** = 60

7.17.1 Detailed Description

7.18 utils Namespace Reference

Classes

- class Classifier
- class Component
- class Embedding
- class FFClassifier
- class Interface
- class Layer
- class Logger
- class ModelWrapper
- class Optimizer
- class ParameterSetter
- class Policy
- class RNNClassifier
- class UNMTPolicy

Functions

- def subclasses (base_cls)
- def copy_dict_hierarchy (dictionary, fill_value=None)
- def merge_dicts (create_dict, iterable)
- def create_leaf_dict (dictionary)
- def reduce_parameters (func, parameters)
- def create_intersection (whole_dict, sub_dict)
- def subtract_dict (whole_dict, sub_dict)
- def **call** (func, iterable, params=None)
- def **format_outputs** (inputs, targets, outputs)
- def ids_from_sentence (vocabulary, sentence)
- def sentence_from_ids (vocabulary, ids)
- def logging (logger)

7.18.1 Detailed Description

7.18.2 Function Documentation

7.18.2.1 def utils.copy_dict_hierarchy (*dictionary, fill_value =* `None`)

```
Copies the a dictionary, that may have multiple embedded dictionaries. The values
of the dictionary are replaced with the provided fill value.
:param dictionary: dict, which structure will be copied.
:param fill_value: the value that will be used to fill the copied dictionary.
:return: dict, the copied dictionary.
```

7.18.2.2 def utils.create_intersection (*whole_dict, sub_dict*)

```
Reduces a dictionary to have the same keys as an another dict.
:param whole_dict: dict, the dictionary to be reduced.
:param sub_dict: dict, the dictionary with the required keys.
:return: dict, the reduced dict.
```

7.18.2.3 def utils.create_leaf_dict (*dictionary*)

```
Creates a flattened dictionary, from a dictionary, that has multiple dictionaries as
values.
:param dictionary: dict, that may have multiple layers of dictionaries as values.
:return: dict, the flattened dictionary.
```

7.18.2.4 def utils.ids_from_sentence (*vocabulary, sentence*)

```
Convenience method, for converting a sequence of words to ids.
:param vocabulary: Language, object of the language to use the look up of.
:param sentence: string, a tokenized sequence of words.
:return: list, containing the ids (int) of the sentence in the same order.
```

7.18.2.5 def utils.logging (*logger*)

```
Decorator for the functions to get logs from. The function which is decorated
with logging must return its values as a dictionary.
:param logger: Logger object, which handles the data.
:return: wrapper of the function.
```

7.18.2.6 def utils.merge_dicts (*create_dict, iterable*)

```
Merges multiple dictionaries, which are created by the output of a function and
an iterable. The function is applied to the values of the iterable, that creates dictionaries
:param create_dict: function, that given a parameter, outputs a dictionary.
:param iterable: iterable, on which the function will be applied.
:return: dict, the created dictionary.
```

7.18.2.7 def utils.reduce_parameters (*func, parameters*)

```
Reduce a set of parameters, given in a form of dictionary, to the
set of parameters, that are required by the function.
:param func: Function, that requires a subset of parameters given as second argument.
:param parameters: dict, a set of parameters, that yields a subset of parameters,
                   that is required by the func parameter.
:return: dict, the subset of parameters for the ’func’ function.
```

7.18.2.8 def utils.sentence_from_ids (*vocabulary, ids*)

```
Convenience method, for converting a sequence of ids to words.
:param vocabulary: Language, object of the language to use the look up of.
:param ids: ids, representations of words.
:return: list, containing the ids (int) of the sentence in the same order.
```

7.18.2.9 def utils.subclasses (*base_cls*)

```
Discovers the inheritance tree of a given class. The class must have an abstract() method. A class of the
hierarchy will only be the part of the returned classes, if it’s abstract() method returns False.
:param base_cls: type, root of the inheritance hierarchy.
:return: dict, classes that are part of the hierarchy, with their str names as keys, and type references as va
```

7.18.2.10 def utils.subtract_dict (*whole_dict,* *sub_dict*)

```
Creates a dictionary with a set of keys, that only present in the ’greater’ dictionary.
:param whole_dict: dict, the dictionary to be reduced.
:param sub_dict: dict, the dictionary with the required keys.
:return: dict, the reduced dict.
```

7.19 validate_vocab Namespace Reference

Functions

- def **validate** (path, dim)
- def **copy** (path_from, path_to, dim)
- def **main** ()

Variables

- string **DEFAULT_INPUT_VOCAB** = '/media/patrik/1EDB65B8599DD93E/data/mapping/vectors-fr-sync.txt'
- string **DEFAULT_OUTPUT_VOCAB** = '/media/patrik/1EDB65B8599DD93E/data/mapping/vectors-fr-sync-valid.txt'
- int **DEFAULT_DIM** = 300

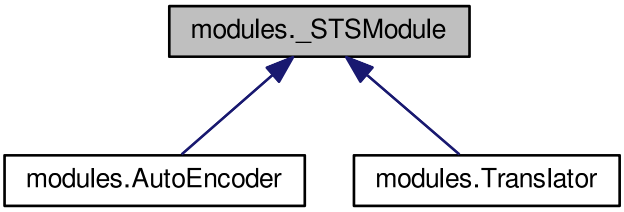
7.19.1 Detailed Description

Chapter 8

Class Documentation

8.1 modules._STSModule Class Reference

Inheritance diagram for modules._STSModule:



Public Member Functions

- def **__init__**
- def **model** (self)
- def **model** (self, model)

8.1.1 Detailed Description

```
A base class for the sequence-to-sequence type modules.
```

The documentation for this class was generated from the following file:

- src/modules/modules.py

8.2 analysis.Analyzer Class Reference

Public Member Functions

- def **__init__** (self, directory)
- def **show_available_metrics** (self)
- def **show_validation_identifiers** (self)
- def **display** (self, metric, mode, plot_size=Plot.PLOT_SIZE, tokens=None, epoch_range=(None,), identifiers=None, kwargs)

Static Public Attributes

- string **TEST_FILE** = 'test.pt'
- string **EVAL_FILE** = 'eval.pt'
- string **OUTPUT_DIR** = 'outputs'
- string **TRAIN_MODE** = 'train'
- string **VALIDATION_MODE** = 'validation'
- string **TEST_MODE** = 'test'
- string **EVALUATION_MODE** = 'evaluation'

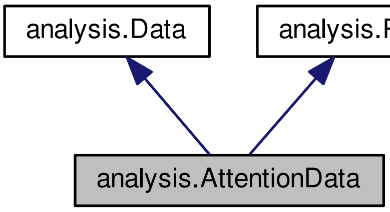
8.2.1 Detailed Description

The documentation for this class was generated from the following file:

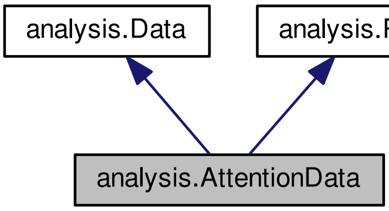
- src/utils/analysis.py

8.3 analysis.AttentionData Class Reference

Inheritance diagram for analysis.AttentionData:



Collaboration diagram for analysis.AttentionData:



Public Member Functions

- def **__init__** (self)
- def **get_required_keys** (self)
- def **add** (self, identifier, value)

Static Public Member Functions

- def **display** (data, plot_size, epochs, epoch_range, identifiers=None, params)

Additional Inherited Members

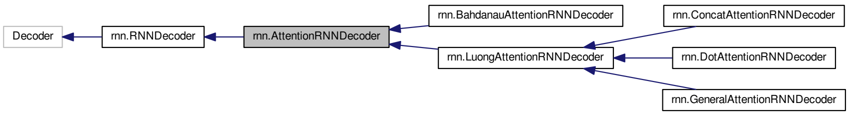
8.3.1 Detailed Description

The documentation for this class was generated from the following file:

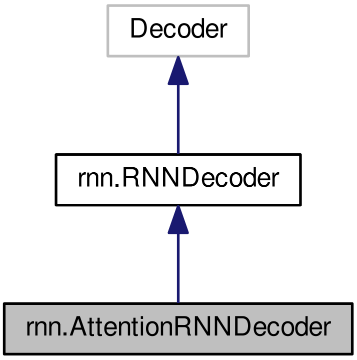
- src/utils/analysis.py

8.4 rnn.AttentionRNNDecoder Class Reference

Inheritance diagram for rnn.AttentionRNNDecoder:


Decoder
rnn.RNNDecoder
rnn.AttentionRNNDecoder
rnn.BahdanauAttentionRNNDecoder
rnn.LuongAttentionRNNDecoder
rnn.ConcatAttentionRNNDecoder
rnn.DotAttentionRNNDecoder
rnn.GeneralAttentionRNNDecoder

Collaboration diagram for rnn.AttentionRNNDecoder:



Public Member Functions

- def **__init__**
- def init_optimizer (self)
- def **forward**
- def **output_types** (self)
- def **optimizers** (self)
- def state (self)
- def **state**

Static Public Attributes

- bool **abstract** = True

Additional Inherited Members

8.4.1 Detailed Description

```
Abstract base class for the attentional variation of recurrent decoder unit.
```

8.4.2 Member Function Documentation

8.4.2.1 def rnn.AttentionRNNDecoder.init_optimizer (*self*, *Decoder*)

```
Initializes the optimizer for the decoder.
```

8.4.2.2 def rnn.AttentionRNNDecoder.state (*self*, *dict*)

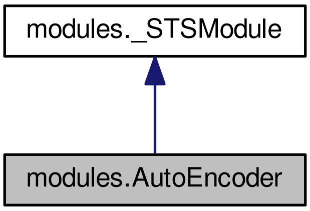
```
Property for the optimizers of the decoder.
```

The documentation for this class was generated from the following file:

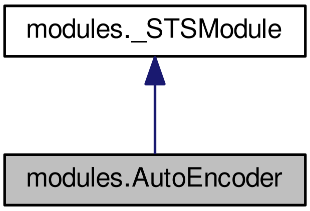
- src/components/decoders/rnn.py

8.5 modules.AutoEncoder Class Reference

Inheritance diagram for modules.AutoEncoder:



Collaboration diagram for modules.AutoEncoder:



Public Member Functions

- def **__init__**
- def **__call__**

8.5.1 Detailed Description

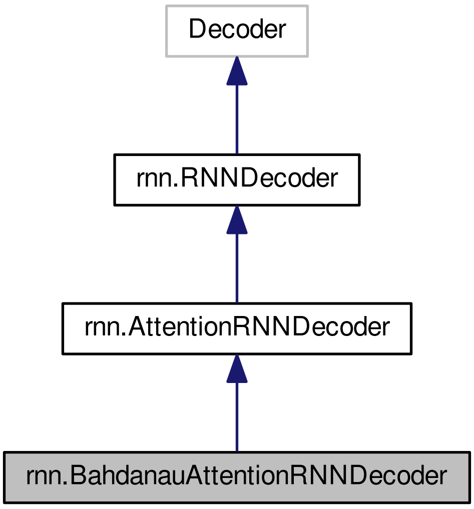
```
Auto-encoder module for a sequence-to-sequence type model.
```

The documentation for this class was generated from the following file:

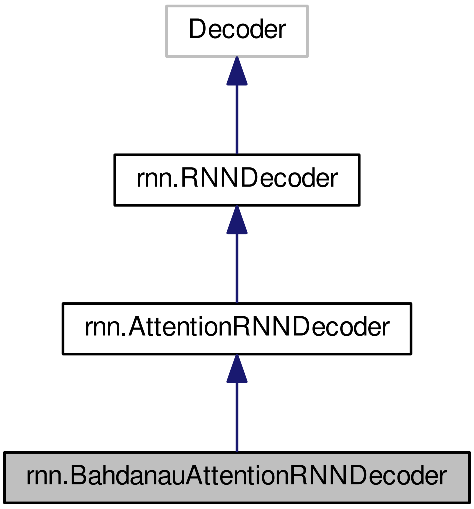
- src/modules/modules.py

8.6 rnn.BahdanauAttentionRNNDecoder Class Reference

Inheritance diagram for rnn.BahdanauAttentionRNNDecoder:



Collaboration diagram for rnn.BahdanauAttentionRNNDecoder:



Public Member Functions

- def **__init__**
- def init_parameters (self)

Static Public Attributes

- **interface**
- bool **abstract** = False

Additional Inherited Members

8.6.1 Detailed Description

```
Global attention mechanism for the recurrent decoder module. The algorithm is based on:

    https://arxiv.org/pdf/1409.0473.pdf

The computational path of the method differs from the Luong style, since
the context vector contributes to the calculation of the hidden state as well, created
by the recurrent unit at time step t.

    h(t-1) -> a(t) -> c(t) -> h(t)

The attention weights are derived from the similarity scores of the previous recurrent hidden
state (from time step t-1) and the encoder outputs. The created context vector is then merged with
the output of the recurrent unit as well, to get the final output of a softmax layer, providing the
probability distribution over the word ids.
```

8.6.2 Member Function Documentation

8.6.2.1 def rnn.BahdanauAttentionRNNDecoder.init_parameters (*self*, *Decoder*)

```
Initializes the parameters for the decoder.
```

8.6.3 Member Data Documentation

8.6.3.1 rnn.BahdanauAttentionRNNDecoder.interface [static]

Initial value:

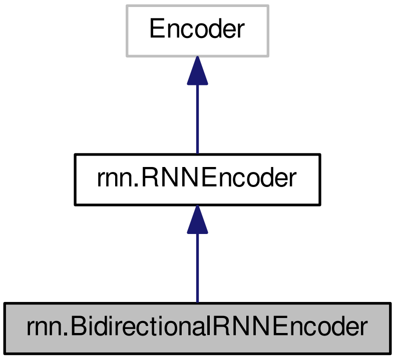
```
= Interface(**{
        **subtract_dict(RNNDecoder.interface.dictionary, {'input_size': None}),
        'input_size': (Interface.last_key(subtract_dict(RNNDecoder.interface.dictionary, {'input_size':
    None})) + 1,
                        'Decoder:embedding_size$ + Decoder:hidden_size$')
    })
```

The documentation for this class was generated from the following file:

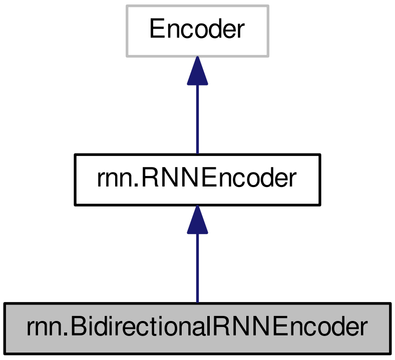
- src/components/decoders/rnn.py

8.7 rnn.BidirectionalRNNEncoder Class Reference

Inheritance diagram for rnn.BidirectionalRNNEncoder:



Collaboration diagram for rnn.BidirectionalRNNEncoder:



Public Member Functions

- def **__init__** (self, parameter_setter)
- def init_parameters (self)
- def forward (self, inputs, lengths)

Static Public Attributes

- **interface** = RNNEncoder.interface
- bool **abstract** = False

Additional Inherited Members

8.7.1 Detailed Description

8.7.2 Member Function Documentation

8.7.2.1 def rnn.BidirectionalRNNEncoder.forward (*self, inputs, lengths*)

```
A forward step of the encoder. The batch of sequences with word ids are
packed into padded_sequence object, which are processed by the recurrent layer.

:param inputs:
    Variable, (batch_size, sequence_length) containing the ids of the words.

:param lengths:
    Ndarray, containing the real lengths of the sequences in the batch (prior to padding).

:return outputs:
    Variable, (batch_size, sequence_length, vocab_size) the output at each time
    step of the encoder.

:return hidden_state:
    Variable, (num_layers * directions, batch_size, hidden_size) the final hidden state.
```

8.7.2.2 def rnn.BidirectionalRNNEncoder.init_parameters (*self*)

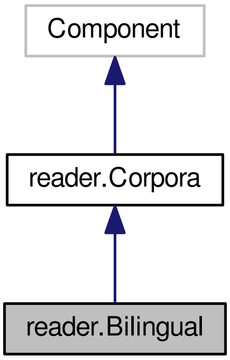
```
Calls the parameter setter, which initializes the Parameter type attributes.
After initialization, the main components of the encoder, which require the previously
initialized parameter values, are created as well.
```

The documentation for this class was generated from the following file:

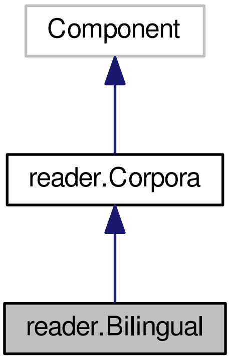
- src/components/encoders/rnn.py

8.8 reader.Bilingual Class Reference

Inheritance diagram for reader.Bilingual:



Collaboration diagram for reader.Bilingual:



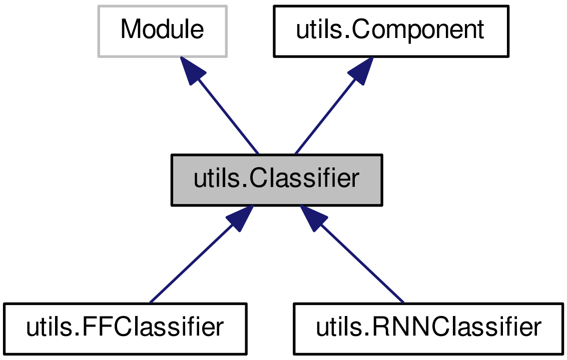
Additional Inherited Members

The documentation for this class was generated from the following file:

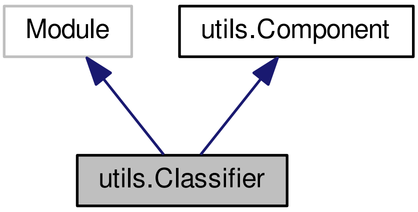
- src/utils/reader.py

8.9 utils.Classifier Class Reference

Inheritance diagram for utils.Classifier:



Collaboration diagram for utils.Classifier:



Public Member Functions

- def **__init__**
- def **forward** (self, args, kwargs)
- def **optimizer** (self)
- def **freeze** (self)
- def **unfreeze** (self)

Static Public Attributes

- **interface**

8.9.1 Detailed Description

```
Abstract base class for the discriminator modules, mainly used for the unsupervised
translation task. Any newly added discriminator type module must inherit from this
super class, otherwise it won’t be discoverable by the hierarchy builder utility.
```

8.9.2 Member Data Documentation

8.9.2.1 utils.Classifier.interface [static]

Initial value:

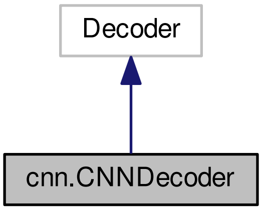
```
= Interface(**{
          ’hidden_size’:       (0, None),
          ’learning_rate’:     (1, None),
          ’optimizer_type’:    (2, None),
          ’output_size’:       (3, None),
          ’cuda’:              (4, ’Experiment:Policy:cuda$’),
          ’input_size’:        (5, ’Encoder:hidden_size$’)
     })
```

The documentation for this class was generated from the following file:

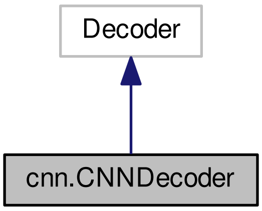
- src/components/utils/utils.py

8.10 cnn.CNNDecoder Class Reference

Inheritance diagram for cnn.CNNDecoder:



Collaboration diagram for cnn.CNNDecoder:

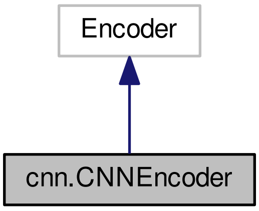


The documentation for this class was generated from the following file:

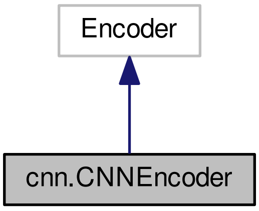
- src/components/decoders/cnn.py

8.11 cnn.CNNEncoder Class Reference

Inheritance diagram for cnn.CNNEncoder:



Collaboration diagram for cnn.CNNEncoder:

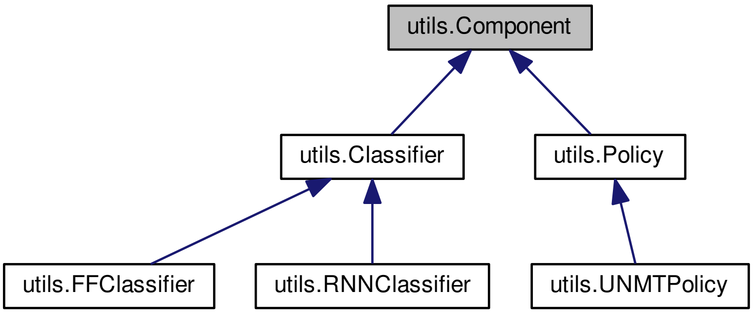


The documentation for this class was generated from the following file:

- src/components/encoders/cnn.py

8.12 utils.Component Class Reference

Inheritance diagram for utils.Component:



Public Member Functions

- def properties (self)

Static Public Attributes

- **interface** = None
- bool **abstract** = True

8.12.1 Detailed Description

```
The base class for the components of the API.
```

8.12.2 Member Function Documentation

8.12.2.1 def utils.Component.properties (*self*)

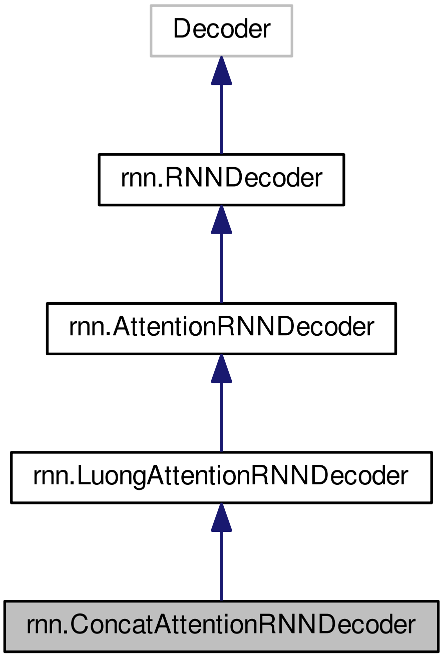
```
Convenience function for retrieving the properties of an instances, with their values.
```

The documentation for this class was generated from the following file:

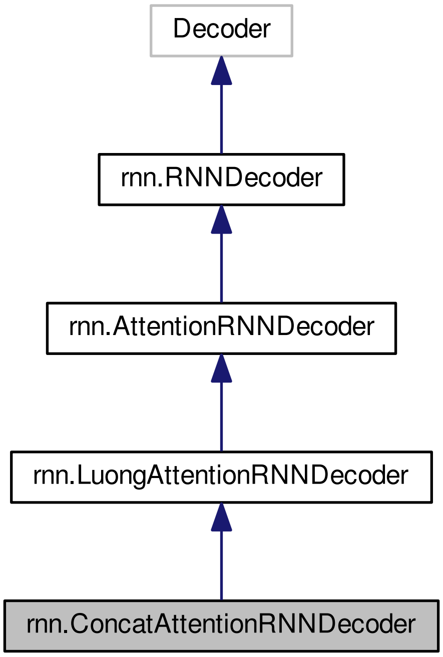
- src/utils/utils.py

8.13 rnn.ConcatAttentionRNNDecoder Class Reference

Inheritance diagram for rnn.ConcatAttentionRNNDecoder:



Collaboration diagram for rnn.ConcatAttentionRNNDecoder:



Public Member Functions

- def **__init__**
- def init_parameters (self)

Static Public Attributes

- **interface** = RNNDecoder.interface
- bool **abstract** = False

Additional Inherited Members

8.13.1 Detailed Description

```
Global attention mechanism for the recurrent decoder module. The algorithm is a specific case
of Luong style attention, where the scoring is based off of the concatenation of the encoder
and decoder states, which is then passed through a non-linear layer with tanh activation.
The result is then multiplied by a vector to transform the final result to the correct size.
The scoring of similarity between encoder and decoder states is essentially the same as Bahdanau's
method, however the computation path follows the Luong style.
```

8.13.2 Member Function Documentation

8.13.2.1 def rnn.ConcatAttentionRNNDecoder.init_parameters (*self,* *Decoder*)

```
Initializes the parameters for the decoder.
```

The documentation for this class was generated from the following file:

- src/components/decoders/rnn.py

8.14 config.Config Class Reference

Public Member Functions

- def **__init__**
- def assemble (self)

8.14.1 Detailed Description

```
Class for handling configurations of models and tasks.
The configs are defined in JSON format files, which are parsed,
and instantiated with the help of interface definitions of the components.
Each node of the JSON file, that has a 'type' and 'params' key are Component type
objects.
```

8.14.2 Member Function Documentation

8.14.2.1 def config.Config.assemble (*self,* *tuple*)

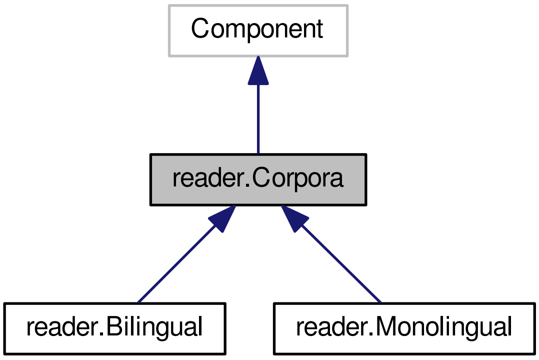
```
Assembles the components, described by the interface of the task from the configuration file.

:return experiment:
    The initialized experiment object.

:return model_dir:
    The location of the experiment outputs.
```

The documentation for this class was generated from the following file:

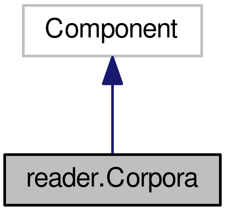
- src/utils/config.py

8.15 reader.Corpora Class Reference

Inheritance diagram for reader.Corpora:



Collaboration diagram for reader.Corpora:



Public Member Functions

- def **__init__**
- def **initialize_corpus** (self)
- def data_path (self)
- def data (self)
- def embedding_size (self)
- def vocabulary (self)
- def **vocabulary**
- def vocab_size (self)

Static Public Attributes

- **interface**
- bool **abstract** = True

8.15.1 Detailed Description

```
Wrapper class for the corpus of the task. Stores information about the corpus, and
stores the location of the train, development and test data.
```

8.15.2 Member Function Documentation

8.15.2.1 def reader.Corpora.data (*self*, *list*)

```
The read data.
```

8.15.2.2 def reader.Corpora.data_path (*self*, *str*)

```
Property for the file path.
```

8.15.2.3 def reader.Corpora.embedding_size (*self*, *int*)

```
Property for the embedding size of the source language.
```

8.15.2.4 def reader.Corpora.vocab_size (*self*, *int*)

```
Property for the vocab size of the language.
```

8.15.2.5 def reader.Corpora.vocabulary (*self*, *Vocabulary*)

```
Property for the vocabularies of the corpora.
```

8.15.3 Member Data Documentation

8.15.3.1 reader.Corpora.interface [static]

Initial value:

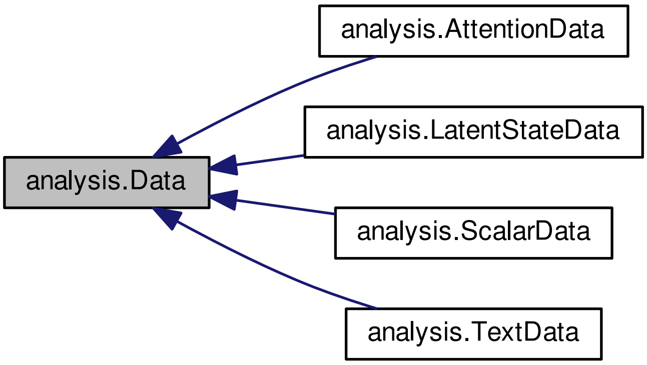
```
= Interface(**{
        'data_path':       (0, None),
        'vocabulary':      (1, ':Vocabulary$'),
        'cuda':            (2, 'Experiment:Policy:cuda$')
    })
```

The documentation for this class was generated from the following file:

- src/utils/reader.py

8.16 analysis.Data Class Reference

Inheritance diagram for analysis.Data:



Public Member Functions

- def **__init__** (self)
- def **add** (self, identifier, value)
- def **get_required_keys** (self)
- def **data** (self)
- def **identifiers** (self)

8.16.1 Detailed Description

The documentation for this class was generated from the following file:

- src/utils/analysis.py

8.17 analysis.DataLog Class Reference

Public Member Functions

- def **__init__** (self, data_interface)
- def **add** (self, identifier, key, value)
- def **get_required_keys** (self, key)
- def **data** (self)
- def **identifiers** (self)

Static Public Attributes

- **str**

8.17.1 Detailed Description

The documentation for this class was generated from the following file:

- src/utils/analysis.py

8.18 analysis.DataLogContainer Class Reference

Public Member Functions

- def **__init__** (self)
- def **add** (self, data_logs)
- def **display** (self, metric, tokens, identifiers, plot_size, epoch_range=(None,), kwargs)
- def **metrics** (self)

8.18.1 Detailed Description

The documentation for this class was generated from the following file:

- src/utils/analysis.py

8.19 reader.DataQueue Class Reference

Public Member Functions

- def **__init__**
- def measure_length (self)
- def generator (self)

Static Public Attributes

- int **MAX_SEGMENT** = 500000

8.19.1 Detailed Description

```
A queue object for the data feed. This can be later configured to load the data to
memory asynchronously.
```

8.19.2 Member Function Documentation

8.19.2.1 def reader.DataQueue.generator (*self*, *list*)

```
Data is retrieved directly from a file, and loaded into data chunks of size MAX_CHUNK_SIZE.
```

8.19.2.2 def reader.DataQueue.measure_length (*self*, *int*)

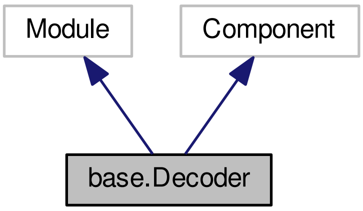
```
Measures the length of the corpora file.
```

The documentation for this class was generated from the following file:

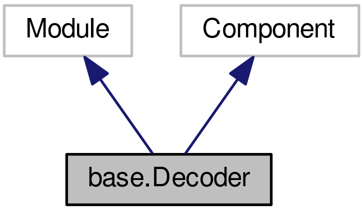
- src/utils/reader.py

8.20 base.Decoder Class Reference

Inheritance diagram for base.Decoder:



Collaboration diagram for base.Decoder:



Public Member Functions

- def **__init__** (self, args, kwargs)
- def **forward** (self, args, kwargs)
- def **optimizers** (self)
- def **state** (self)
- def **state** (self, value)
- def **output_size** (self)

8.20.1 Detailed Description

```
Abstract base class for the decoder modules of the application. A decoder must
inherit from this class, otherwise it won't be discoverable by the hierarchy
builder utility.
```

The documentation for this class was generated from the following file:

- src/components/base.py

8.21 modules.Discriminator Class Reference

Public Member Functions

- def **__init__**
- def __call__ (self, args, inputs, targets, kwargs)

8.21.1 Detailed Description

8.21.2 Member Function Documentation

8.21.2.1 def modules.Discriminator.__call__ (*self, args, inputs, targets, kwargs*)

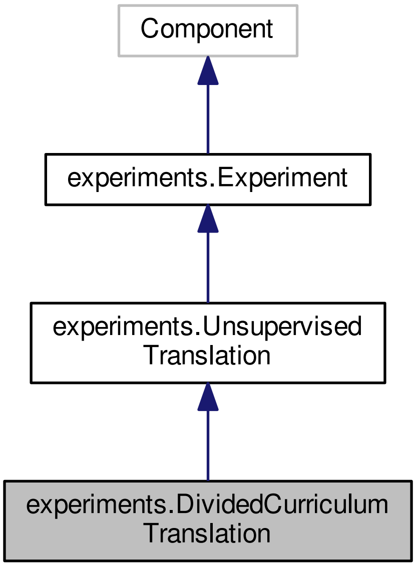
```
This function implements the discrimination mechanism, where the inputs and the targets - which
are required for the evaluation and training of the discriminator - are created. The inputs are
fed into the discriminator and evaluated based on the cross entropy loss, that is defined in the
init function. The targets are either one-hot coded vectors, or their inverse. This depends on
whether the loss is calculated for the discriminator or model loss.

:param inputs:
    A list, containing the batches from the input pipelines.

:param targets:
    An int value, that represents the index of the encoder's input language.

:param lengths:

:return loss:
    A scalar loss value, indicating the average loss of the discriminator for either the
    inverse or normal target vector.
```

The documentation for this class was generated from the following file:

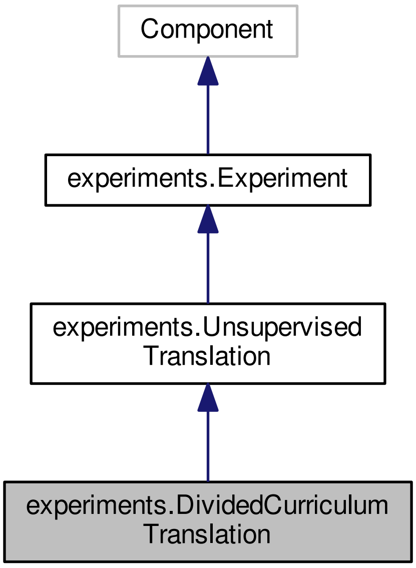
- src/modules/modules.py

8.22 experiments.DividedCurriculumTranslation Class Reference

Inheritance diagram for experiments.DividedCurriculumTranslation:


Component
experiments.Experiment
experiments.Unsupervised Translation
experiments.DividedCurriculum Translation

Collaboration diagram for experiments.DividedCurriculumTranslation:


Component
experiments.Experiment
experiments.Unsupervised Translation
experiments.DividedCurriculum Translation

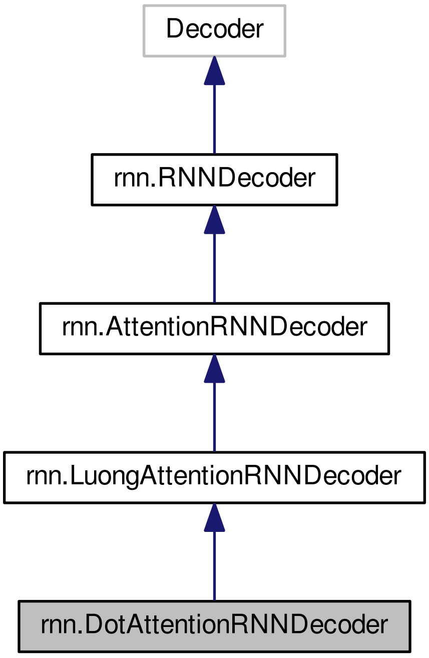
Additional Inherited Members

The documentation for this class was generated from the following file:

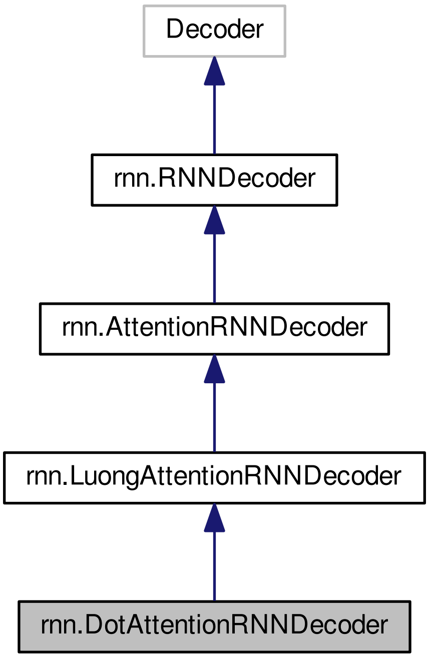
- src/experiments/experiments.py

8.23 rnn.DotAttentionRNNDecoder Class Reference

Inheritance diagram for rnn.DotAttentionRNNDecoder:



Collaboration diagram for rnn.DotAttentionRNNDecoder:



Public Member Functions

- def **__init__**
- def init_parameters (self)

Static Public Attributes

- **interface** = RNNDecoder.interface
- bool **abstract** = False

Additional Inherited Members

8.23.1 Detailed Description

```
Global attention mechanism for the recurrent decoder module. The algorithm is a specific case
of Luong style attention, where the scoring is based off of only the dot product of the
encoder and decoder states.
```

8.23.2 Member Function Documentation

8.23.2.1 def rnn.DotAttentionRNNDecoder.init_parameters (*self*, *Decoder*)

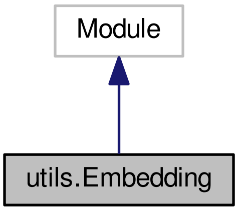
```
Initializes the parameters for the decoder.
```

The documentation for this class was generated from the following file:

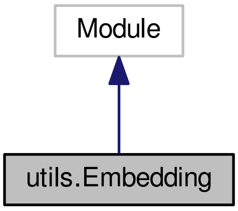
- src/components/decoders/rnn.py

8.24 utils.Embedding Class Reference

Inheritance diagram for utils.Embedding:



Collaboration diagram for utils.Embedding:



Public Member Functions

- def **__init__**
- def forward (self, inputs)
- def freeze (self)
- def unfreeze (self)
- def optimizer (self)
- def state (self)
- def state (self, states)

8.24.1 Detailed Description

```
Wrapper class for the embedding layers of the models. The optional training of the embeddings
is done by a built-in optimizer.
```

8.24.2 Member Function Documentation

8.24.2.1 def utils.Embedding.forward (*self, inputs*)

```
Propagates the inputs through the embedding layer.

:param inputs:
    Variable, word id-s, that will be translated to word vector representations.

:return outputs:
    Variable, word vectors of the given input.
```

8.24.2.2 def utils.Embedding.freeze (*self*)

```
Freezes the parameters of the embedding layer. While frozen, the parameters can’t be
modified by the optimizer.
```

8.24.2.3 def utils.Embedding.optimizer (*self*)

```
Property for the optimizer of the embedding.
```

8.24.2.4 def utils.Embedding.state (*self*)

```
Property for the state of the embedding.
```

8.24.2.5 def utils.Embedding.state (*self, states*)

```
Setter method for the state of the embedding.
```

8.24.2.6 def utils.Embedding.unfreeze (*self*)

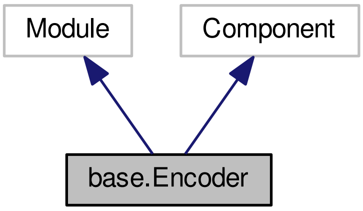
```
       Unfreezes the parameters of the optimizers. By calling this method, the optimizer will
       be able to modify the weights of the embedding layer.
```

The documentation for this class was generated from the following file:

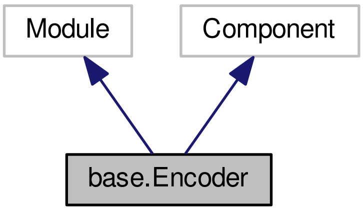
- src/components/utils/utils.py

8.25 base.Encoder Class Reference

Inheritance diagram for base.Encoder:



Collaboration diagram for base.Encoder:



Public Member Functions

- def **__init__** (self, args, kwargs)
- def **forward** (self, args, kwargs)
- def **optimizers** (self)
- def **state** (self)
- def **state** (self, value)

8.25.1 Detailed Description

```
Abstract base class for the encoder modules of the application. An encoder must
inherit from this class, otherwise it won't be discoverable by the hierarchy
builder utility.
```

The documentation for this class was generated from the following file:

- src/components/base.py

8.26 session.EvaluationContext Class Reference

Public Member Functions

- def **__init__** (self, session)
- def **__enter__** (self)
- def **__exit__** (self, exc_type, exc_val, exc_tb)
- def **evaluate** (self)
- def **save** (self)

Static Public Attributes

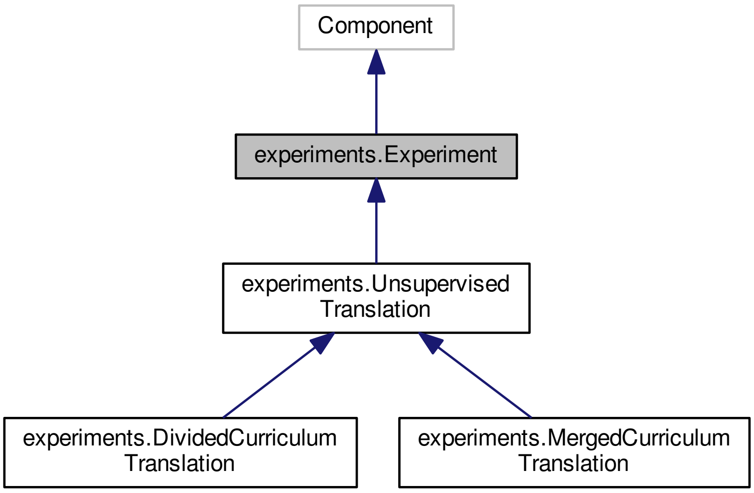
- string **EVAL_FILE** = 'eval.pt'

The documentation for this class was generated from the following file:

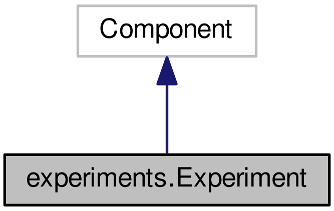
- src/utils/session.py

8.27 experiments.Experiment Class Reference

Inheritance diagram for experiments.Experiment:


Component
experiments.Experiment
experiments.Unsupervised Translation
experiments.DividedCurriculum Translation
experiments.MergedCurriculum Translation

Collaboration diagram for experiments.Experiment:



Public Member Functions

- def **train** (self, epoch)
- def **validate** (self)
- def **test** (self)
- def **evaluate** (self)
- def **state** (self)
- def **state** (self, value)

8.27.1 Detailed Description

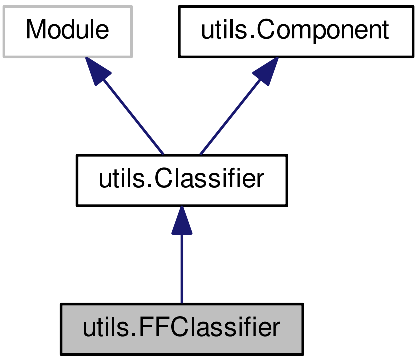
```
Abstract base class for the experiments.
```

The documentation for this class was generated from the following file:

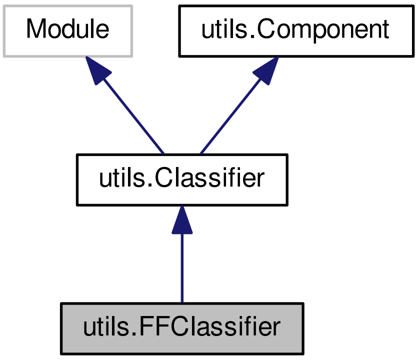
- src/experiments/experiments.py

8.28 utils.FFClassifier Class Reference

Inheritance diagram for utils.FFClassifier:



Collaboration diagram for utils.FFClassifier:



Public Member Functions

- def **__init__**
- def forward (self, args, inputs, kwargs)
- def **optimizer** (self)

Static Public Attributes

- bool **abstract** = False

8.28.1 Detailed Description

```
Feed-forward classifier module for the unsupervised neural translation task.
```

8.28.2 Member Function Documentation

8.28.2.1 def utils.FFClassifier.forward (*self, args, inputs, kwargs*)

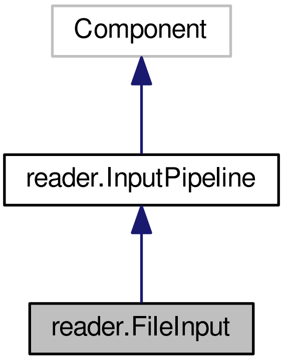
```
Forward step for the classifier.

:param inputs:
    Variable, (batch_size, input_size), where input_size is equal to the encoder’s
    hidden_size.

:return output:
    Variable, (batch_size, 1).
```

The documentation for this class was generated from the following file:

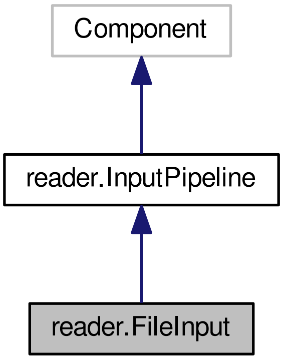
- src/components/utils/utils.py

8.29 reader.FileInput Class Reference

Inheritance diagram for reader.FileInput:



Collaboration diagram for reader.FileInput:



Public Member Functions

- def **__init__**
- def batch_generator (self)
- def corpora (self)
- def vocabulary (self)
- def **batch_size** (self)

Public Attributes

- **total_length**

Static Public Attributes

- **interface**
- bool **abstract** = False

8.29.1 Detailed Description

```
An implementation of the reader class. Batches are read from the source in file real-time.
This version of the reader should only be used if the source file is too large to be stored
in memory.
```

8.29.2 Member Function Documentation

8.29.2.1 def reader.FileInput.batch_generator (*self*)

```
Generator for mini-batches. Data is read from memory. The _format_batch function comes from the
definition of the task. It is a wrapper function that transform the generated batch of data into a form,
that is convenient for the current task.

:return:
    tuple, a PyTorch Variable of dimension (Batch_size, Sequence_length), containing
    the ids of words, sorted by their length in descending order. Each sample is
    padded to the length of the longest sequence in the batch/segment.
    The latter behaviour may vary. Second element of the tuple is a numpy array
    of the lengths of the original sequences (without padding).
```

8.29.2.2 def reader.FileInput.corpora (*self*)

```
Property for the corpora of the reader.
```

8.29.2.3 def reader.FileInput.vocabulary (*self*)

```
Property for the reader object's vocabulary.
```

8.29.3 Member Data Documentation

8.29.3.1 reader.FileInput.interface [static]

Initial value:

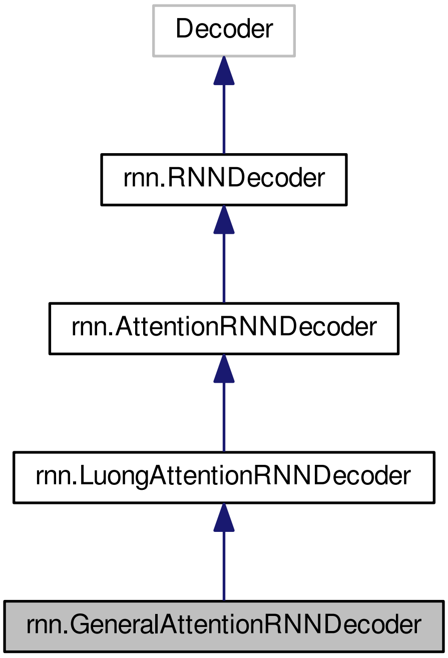
```
= Interface(**{
        'max_segment_size':     (0, None),
        'batch_size':           (1, None),
        'padding_type':         (2, None),
        'shuffle':              (3, None),
        'cuda':                 (4, 'Experiment:Policy:cuda$'),
        'corpora':              (5, Corpora)
    })
```

The documentation for this class was generated from the following file:

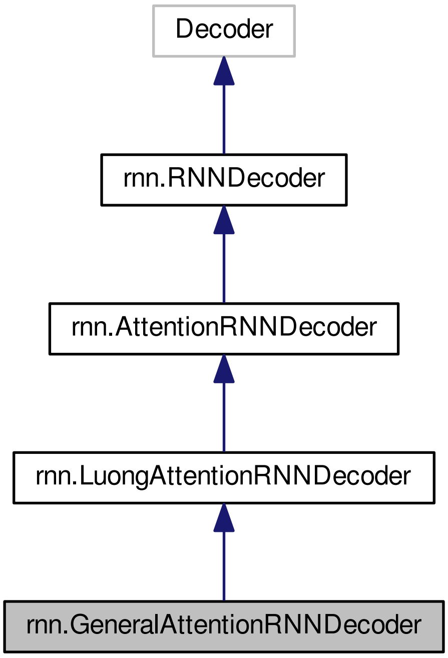
- src/utils/reader.py

8.30 rnn.GeneralAttentionRNNDecoder Class Reference

Inheritance diagram for rnn.GeneralAttentionRNNDecoder:



Collaboration diagram for rnn.GeneralAttentionRNNDecoder:



Public Member Functions

- def **__init__**
- def init_parameters (self)

Static Public Attributes

- **interface** = RNNDecoder.interface
- bool **abstract** = False

Additional Inherited Members

8.30.1 Detailed Description

```
Global attention mechanism for the recurrent decoder module. The algorithm is a specific case
of Luong style attention, where the scoring is based off of the linear activation
of the encoder output, and the dot product of the decoder hidden state with the result of the
activation from the linear layer.
```

8.30.2 Member Function Documentation

8.30.2.1 def rnn.GeneralAttentionRNNDecoder.init_parameters (*self*, *Decoder*)

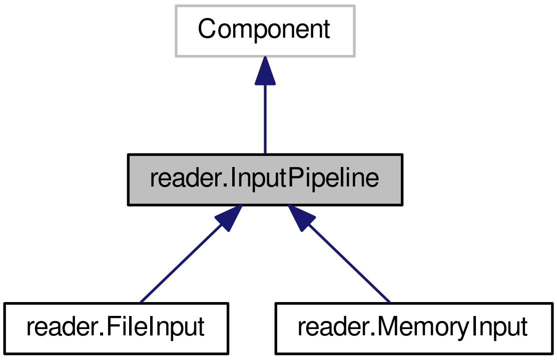
```
Initializes the parameters for the decoder.
```

The documentation for this class was generated from the following file:

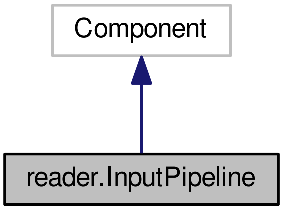
- src/components/decoders/rnn.py

8.31 reader.InputPipeline Class Reference

Inheritance diagram for reader.InputPipeline:



Collaboration diagram for reader.InputPipeline:



Public Member Functions

- def batch_generator (self)

8.31.1 Detailed Description

```
Derived classes should implement the reading logic for the seq2seq model. Readers divide the
data into segments. The purpose of this behaviour, is to keep the sentences with similar lengths
in segments, so they can be freely shuffled without mixing them together with larger sentences.
```

8.31.2 Member Function Documentation

8.31.2.1 def reader.InputPipeline.batch_generator (*self*)

```
The role of this function is to generate batches for the seq2seq model. The batch generation
should include the logic of shuffling the samples. A full iteration should include
all of the data samples.
```

The documentation for this class was generated from the following file:

- src/utils/reader.py

8.32 utils.Interface Class Reference

Public Member Functions

- def **__init__** (self, kwargs)
- def **__getitem__** (self, key)
- def **__next__** (self)
- def **__iter__** (self)
- def **dictionary** (self)

Static Public Member Functions

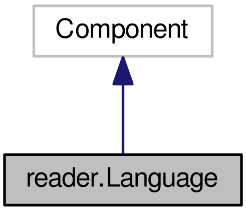
- def **last_key** (dictionary)

The documentation for this class was generated from the following file:

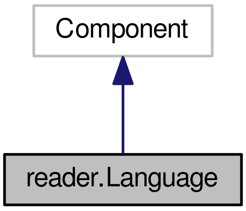
- src/utils/utils.py

8.33 reader.Language Class Reference

Inheritance diagram for reader.Language:



Collaboration diagram for reader.Language:



Public Member Functions

- def **__init__**
- def **identifier** (self)
- def **input_pipelines** (self)
- def **vocabulary** (self)

Static Public Attributes

- bool **abstract** = False
- **interface**

8.33.1 Detailed Description

```
An abstract representation of ta language in an experiment. This class holds all relevant
information about a given language, its vocabulary, identifier and the corpus.
```

8.33.2 Member Data Documentation

8.33.2.1 reader.Language.interface [static]

Initial value:

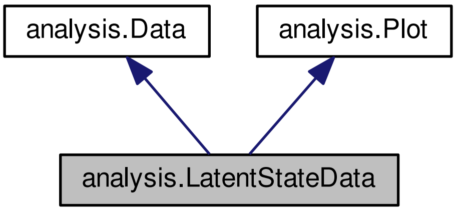
```
= Interface(**{
        'identifier':           (0, None),
        'vocabulary':           (1, Vocabulary),
        'input_pipelines':      (2, InputPipeline),
    })
```

The documentation for this class was generated from the following file:

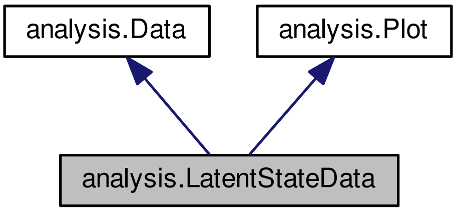
- src/utils/reader.py

8.34 analysis.LatentStateData Class Reference

Inheritance diagram for analysis.LatentStateData:



Collaboration diagram for analysis.LatentStateData:



Public Member Functions

- def **__init__** (self)
- def **add** (self, identifier, value)
- def **get_required_keys** (self)

Static Public Member Functions

- def **display** (data, plot_size, epochs, epoch_range, identifiers=None, params)

Additional Inherited Members

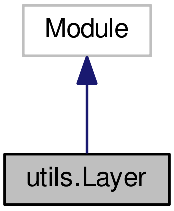
8.34.1 Detailed Description

The documentation for this class was generated from the following file:

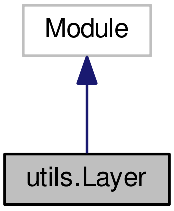
- src/utils/analysis.py

8.35 utils.Layer Class Reference

Inheritance diagram for utils.Layer:



Collaboration diagram for utils.Layer:



Public Member Functions

- def __init__ (self, input_size, output_size, use_cuda)
- def forward (self, inputs)
- def **freeze** (self)
- def **unfreeze** (self)
- def optimizer (self)
- def state (self)
- def state (self, states)

Public Attributes

- **size**

8.35.1 Constructor & Destructor Documentation

8.35.1.1 def utils.Layer.__init__ (*self, input_size, output_size, use_cuda*)

```
:param input_size:

:param output_size:

:param use_cuda:
```

8.35.2 Member Function Documentation

8.35.2.1 def utils.Layer.forward (*self, inputs*)

```
:param inputs:

:return outputs:
```

8.35.2.2 def utils.Layer.optimizer (*self*)

```
Property for the optimizer of the layer.
```

8.35.2.3 def utils.Layer.state (*self*)

```
Property for the state of the embedding.
```

8.35.2.4 def utils.Layer.state (*self, states*)

```
Setter method for the state of the embedding.
:param states: dict, containing the state of the weights and optimizer.
```

The documentation for this class was generated from the following file:

- src/components/utils/utils.py

8.36 utils.Logger Class Reference

Public Member Functions

- def __init__ (self, params, dump_interval=1000)
- def __call__ (self, args, func, kwargs)
- def log_dir (self)
- def log_dir (self, log_dir)

8.36.1 Detailed Description

```
Logger class for saving the progress of training.
```

8.36.2 Constructor & Destructor Documentation

8.36.2.1 def utils.Logger.__init__ (*self*, *params*, *dump_interval* = 1000)

```
A logger instance. Instantiation should happen as a parameter of logging decorator.
:param params: tuple, name of the input parameters, which will be logged.
:param dump_interval: int, number of iteration, between two log dumps.
```

8.36.3 Member Function Documentation

8.36.3.1 def utils.Logger.__call__ (*self*, *args*, *func*, *kwargs*)

```
Invocation of a logger object will execute the given function, record the time required
for this operation, and then save the results and given input parameters to the log dictionary.
:param args: arguments of the function, which will be executed.
:param func: function to be executed.
:param kwargs: keyword arguments of the function to be executed.
:return: result of the execution.
```

8.36.3.2 def utils.Logger.log_dir (*self*)

```
Property for the logging directory.
:return: str, location of the logs.
```

8.36.3.3 def utils.Logger.log_dir (*self*, *log_dir*)

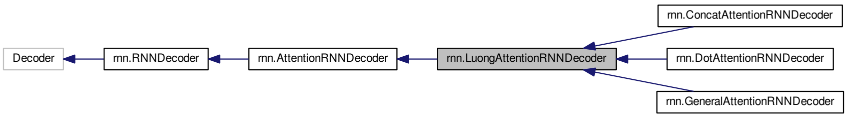
```
Setter for the directory of the logging directory.
```

The documentation for this class was generated from the following file:

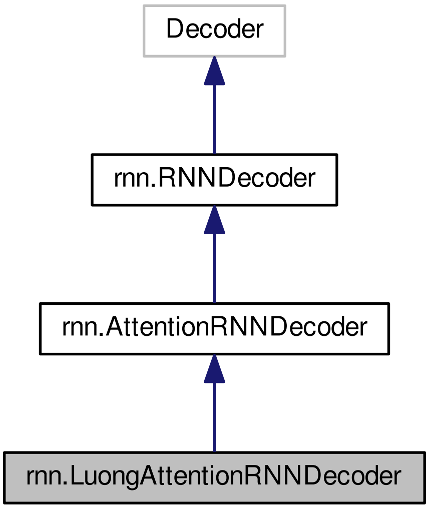
- src/utils/utils.py

8.37 rnn.LuongAttentionRNNDecoder Class Reference

Inheritance diagram for rnn.LuongAttentionRNNDecoder:


Decoder
rnn.RNNDecoder
rnn.AttentionRNNDecoder
rnn.LuongAttentionRNNDecoder
rnn.ConcatAttentionRNNDecoder
rnn.DotAttentionRNNDecoder
rnn.GeneralAttentionRNNDecoder

Collaboration diagram for rnn.LuongAttentionRNNDecoder:



Public Member Functions

- def **__init__**
- def init_parameters (self)

Additional Inherited Members

8.37.1 Detailed Description

```
Attention mechanism for the recurrent decoder module. The algorithm is based on:

    https://arxiv.org/pdf/1508.04025.pdf

The computational path of the method differs from the Bahdanau style, since
here the context vector contributes to the calculation of the hidden state, after the
computations of the recurrent layer.

    h(t) -> a(t) -> c(t) -> h*(t)
```

8.37.2 Member Function Documentation

8.37.2.1 def rnn.LuongAttentionRNNDecoder.init_parameters (*self*, *Decoder*)

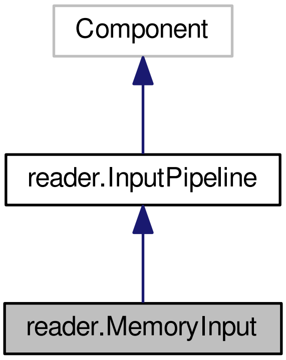
```
Initializes the parameters for the decoder.
```

The documentation for this class was generated from the following file:

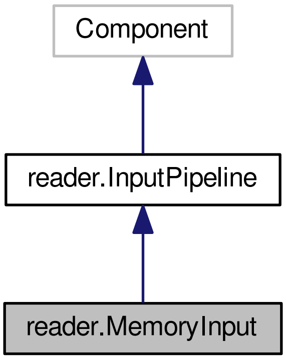
- src/components/decoders/rnn.py

8.38 reader.MemoryInput Class Reference

Inheritance diagram for reader.MemoryInput:



Collaboration diagram for reader.MemoryInput:



Public Member Functions

- def **__init__**
- def batch_generator (self)
- def print_validation_format (self, dictionary)
- def corpora (self)
- def vocabulary (self)
- def **batch_size** (self)

Static Public Attributes

- **interface**
- bool **abstract** = False

8.38.1 Detailed Description

```
A faster implementation of reader class than FileReader. The source data is fully loaded into
the memory.
```

8.38.2 Member Function Documentation

8.38.2.1 def reader.MemoryInput.batch_generator (*self*)

```
Generator for mini-batches. Data is read from memory. The _format_batch function comes from the
definition of the task. It is a wrapper function that transform the generated batch of data into a form,
that is convenient for the current task.

:return:
    tuple, a PyTorch Variable of dimension (Batch_size, Sequence_length), containing
    the ids of words, sorted by their length in descending order. Each sample is
    padded to the length of the longest sequence in the batch/segment.
    The latter behaviour may vary. Second element of the tuple is a numpy array
    of the lengths of the original sequences (without padding).
```

8.38.2.2 def reader.MemoryInput.corpora (*self*)

```
Property for the corpora of the reader.
```

8.38.2.3 def reader.MemoryInput.print_validation_format (*self,* *dictionary*)

```
Convenience function for printing the parameters of the function, to the standard output.
The parameters must be provided as keyword arguments. Each argument must contain a 2D
array containing word ids, which will be converted to the represented words from the
dictionary of the language, used by the reader instance.
```

8.38.2.4 def reader.MemoryInput.vocabulary (*self*)

```
Property for the reader object’s vocabulary.
```

8.38.3 Member Data Documentation

8.38.3.1 reader.MemoryInput.interface [static]

Initial value:

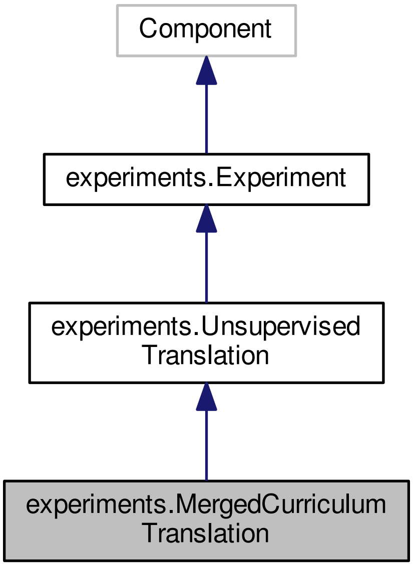
```
= Interface(**{
        ’max_segment_size’:  (0, None),
        ’batch_size’:        (1, None),
        ’padding_type’:      (2, None),
        ’shuffle’:           (3, None),
        ’cuda’:              (4, ’Experiment:Policy:cuda$’),
        ’corpora’:           (5, Corpora)
    })
```

The documentation for this class was generated from the following file:

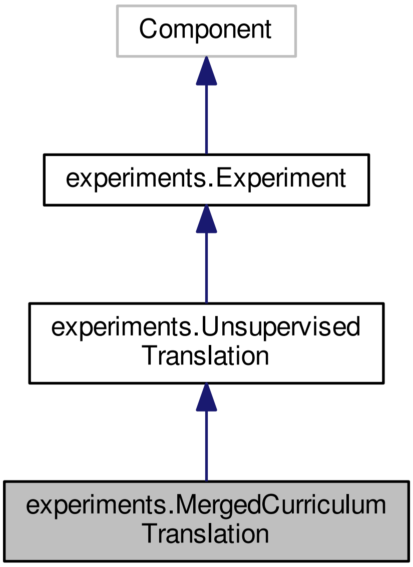
- src/utils/reader.py

8.39 experiments.MergedCurriculumTranslation Class Reference

Inheritance diagram for experiments.MergedCurriculumTranslation:


Component
experiments.Experiment
experiments.Unsupervised Translation
experiments.MergedCurriculum Translation

Collaboration diagram for experiments.MergedCurriculumTranslation:



Public Member Functions

- def __**init**__
- def **train**
- def validate (self)
- def test (self)
- def evaluate (self)
- def state (self)
- def state (self, state)

Public Attributes

- **reguralize**

Static Public Attributes

- **interface** = UnsupervisedTranslation.interface
- bool **abstract** = False

Additional Inherited Members

8.39.1 Detailed Description

8.39.2 Member Function Documentation

8.39.2.1 def experiments.MergedCurriculumTranslation.evaluate (*self*, *dict*)

```
This function evaluates the model. Input data is propagated forward, and then the loss calculated
based on the same loss function which was used during training. The weights however, are not modified
in this function.

:return logs:
    A list of DataLog type objects, that contain the logging data for the languages. The number of
    data logs equal to the number of languages, and each data log contains information about the
    produced output for the whole data set of a language.

    Additional outputs depend on the chosen model.
```

8.39.2.2 def experiments.MergedCurriculumTranslation.state (*self*)

```
Property for the state of the task.
```

8.39.2.3 def experiments.MergedCurriculumTranslation.state (*self*, *state*)

```
Setter function for the state of the task, and the embeddings.
```

8.39.2.4 def experiments.MergedCurriculumTranslation.test (*self*, *dict*)

```
This function evaluates the model. Input data is propagated forward, and then the loss calculated
based on the same loss function which was used during training. The weights however, are not modified
in this function.

:return logs:
    A list of DataLog type objects, that contain the logging data for the languages. The number of
    data logs equal to the number of languages, and each data log contains information about the
    produced output for the whole data set of a language.

    Additional outputs depend on the chosen model.
```

8.39.2.5 def experiments.MergedCurriculumTranslation.validate (*self*, *dict*)

```
This function evaluates the model. Input data is propagated forward, and then the loss calculated
based on the same loss function which was used during training. The weights however, are not modified
in this function.

:return logs:
    A list of DataLog type objects, that contain the logging data for the languages. The number of
    data logs equal to the number of languages, and each data log contains information about the
    produced output for the whole data set of a language.

total_loss:
    The total loss of the iteration, which is the same as the model loss during training.
    The value contains the loss of translation, auto-encoding and reguralization loss. The
    individual error of the discriminator is not included.

translation_loss:
    The error, that is produced by the model, when translating a sentence.
```

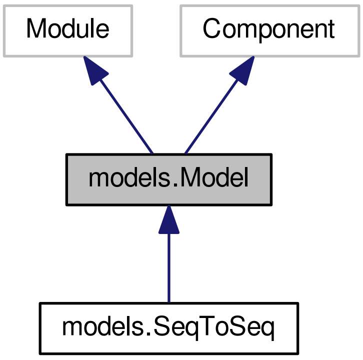
```
auto_encoding_loss:
    The error, that is produced by the model,
    when restoring (auto-encoding) a sentence.

reguralization_loss:
    The reguralization loss, that is produced by the discriminator.

discriminator_loss:
    The error of the discriminator, which is the loss that is produced, when the
    discriminator identifies a given latent vector.

translation_text:
    The textual representation of the input, target and output symbols at the
    translation phase. These texts are produced by the format outputs
    utility function.

auto_encoding_text:
    The textual representation of the input, target and output symbols at the
    auto encoding phase. These texts are produced by the format outputs
    utility function.

    Additional outputs depend on the chosen model.
```

The documentation for this class was generated from the following file:

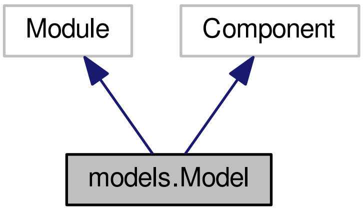
- src/experiments/experiments.py

8.40 models.Model Class Reference

Inheritance diagram for models.Model:



Collaboration diagram for models.Model:



Public Member Functions

- def **__init__** (self, args, kwargs)
- def **forward** (self, args, kwargs)
- def **optimizers** (self)
- def **state** (self)
- def **state** (self, value)

8.40.1 Detailed Description

```
Abstract base class for the models of the application.
```

The documentation for this class was generated from the following file:

- src/models/models.py

8.41 utils.ModelWrapper Class Reference

Public Member Functions

- def **__init__** (self, model, tokens)
- def __call__ (self, args, kwargs)
- def **init_table** (self, lookups)
- def **switch_lookups** (self, lookups)
- def set_lookup (self, lookups)
- def **encoder** (self)

8.41.1 Member Function Documentation

8.41.1.1 def utils.ModelWrapper.__call__ (*self, args, kwargs*)

```
Forwards the call to the actual model.
```

8.41.1.2 def utils.ModelWrapper.set_lookup (*self*, *lookups*)

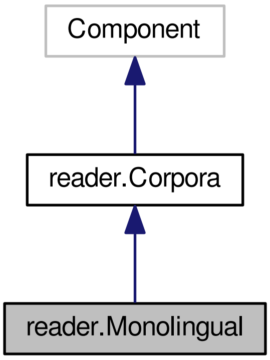
```
Sets the lookups (embeddings) for the encoder and decoder.

Arguments:
    lookups:
A dictionary, that yields the new embeddings for the decoder and encoder.
The dictionary has to contain 3 keys, E_I, D_I, and D_O. The values of the keys
are ints, which represent the index of the languages.
```

The documentation for this class was generated from the following file:

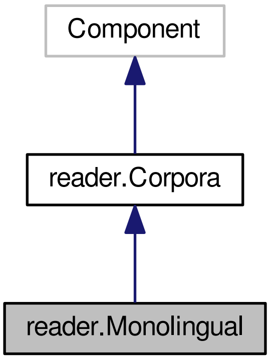
- src/utils/utils.py

8.42 reader.Monolingual Class Reference

Inheritance diagram for reader.Monolingual:



Collaboration diagram for reader.Monolingual:



Public Member Functions

- def **__init__**
- def initialize_corpus (self)

Static Public Attributes

- **interface** = Corpora.interface
- bool **abstract** = False

8.42.1 Detailed Description

```
Special case of Corpora class, where the data read from the files only have a single language.
```

8.42.2 Member Function Documentation

8.42.2.1 def reader.Monolingual.initialize_corpus (*self*)

The documentation for this class was generated from the following file:

- src/utils/reader.py

8.43 modules.NoiseModel Class Reference

Public Member Functions

- def **__init__** (self, use_cuda, p=0.1, k=3)
- def **__call__** (self, inputs, padding)

The documentation for this class was generated from the following file:

- src/modules/modules.py

8.44 utils.Optimizer Class Reference

Public Member Functions

- def __init__ (self, parameters, optimizer_type, scheduler_type, learning_rate)
- def step (self)
- def clear (self)
- def adjust (self, metric)
- def state (self)
- def state (self, state)

8.44.1 Detailed Description

```
Wrapper class for the optimizers. Additionally to the optimizers provided by torch,
this type has built-in learning rate scheduling.
```

8.44.2 Constructor & Destructor Documentation

8.44.2.1 def utils.Optimizer.__init__ (*self, parameters, optimizer_type, scheduler_type, learning_rate*)

```
Optimizer type object.

:param parameters:
    Iterable, containing the parameters, that will be optimized by the provided
    optimalization algorithm.

:param optimizer_type:
    Str, type of the algorithm to be used for optimalization.

:param scheduler_type:
    Str, type of the scheduler to be used for learning rate adjustments.

:param learning_rate:
    Float, the initial learning rate.
```

8.44.3 Member Function Documentation

8.44.3.1 def utils.Optimizer.adjust (*self, metric*)

```
Adjust the learning rate, given a metric.
```

8.44.3.2 def utils.Optimizer.clear (*self*)

```
Clears the gradients of the parameters, which are being optimized by the algorithm.
```

8.44.3.3 def utils.Optimizer.state (*self*)

```
Property for the state of the optimizer.
```

8.44.3.4 def utils.Optimizer.state (*self, state*)

```
Setter method for the state of the optimizer.
```

8.44.3.5 def utils.Optimizer.step (*self*)

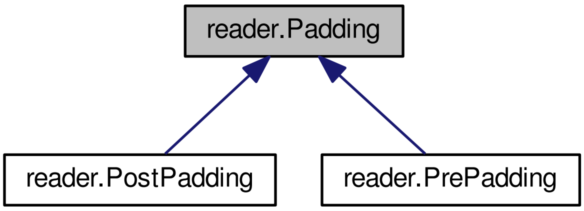
```
Executes the optimalization step on the parameters, that have benn provided to the optimizer.
```

The documentation for this class was generated from the following file:

- src/components/utils/utils.py

8.45 reader.Padding Class Reference

Inheritance diagram for reader.Padding:



Public Member Functions

- def **__init__**
- def **create_batch** (self, data)

Static Public Attributes

- bool **abstract** = True

8.45.1 Detailed Description

```
Base class for the padding types.
```

The documentation for this class was generated from the following file:

- src/utils/reader.py

8.46 reader.ParallelDataQueue Class Reference

The documentation for this class was generated from the following file:

- src/utils/reader.py

8.47 utils.ParameterSetter Class Reference

Public Member Functions

- def __init__ (self, param_dict)
- def initialize (self, instance, subset=None)
- def extract (self, parameter_dict)

Static Public Member Functions

- def pack (cls_interface)

8.47.1 Detailed Description

```
This class handles the initialization of the given object's parameters.
```

8.47.2 Constructor & Destructor Documentation

8.47.2.1 def utils.ParameterSetter.__init__ (*self*, *param_dict*)

```
An instance of a ParameterSetter class.
:param param_dict: dict, containing the key value pairs of the object's parameters.
```

8.47.3 Member Function Documentation

8.47.3.1 def utils.ParameterSetter.extract (*self*, *parameter_dict*)

```
Extracts a set of parameters from the parameter dictionary of the setter object. The extracted parameters
are then removed from the parameter setter's dictionary.
:param parameter_dict: dict, subset of parameters to be extracted. It is typically an interface of an instance
:return: dict, extracted parameters.
```

8.47.3.2 def utils.ParameterSetter.initialize (*self*, *instance*, *subset* = None)

```
This function creates the attributes of an instances. The instance must have an interface() method,
that describes the required attributes.
:param instance: instance, that will be initialized with the parameters, stored in the parameter dict.
:param subset: dict, subset of the parameter setter's param dict, that specifies the subset
       that should be created for the instance. If None, all of the parameters will be
       initialized, that are stored in the param dict.
```

8.47.3.3 def utils.ParameterSetter.pack (*cls_interface*) [static]

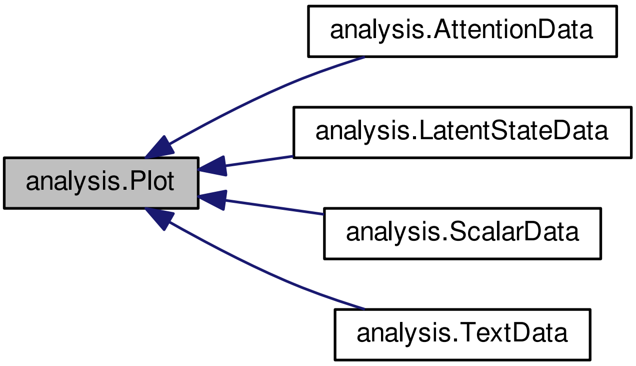
```
Packs the parameters of the decorated function.
:param cls_interface: dict, interface of the instance.
```

The documentation for this class was generated from the following file:

- src/utils/utils.py

8.48 analysis.Plot Class Reference

Inheritance diagram for analysis.Plot:



Static Public Member Functions

- def **display** (data, plot_size, epochs, epoch_range, identifiers=None, params)

Static Public Attributes

- int **PLOT_SIZE** = 8

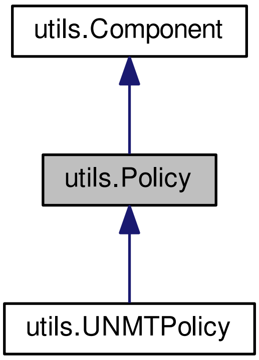
8.48.1 Detailed Description

The documentation for this class was generated from the following file:

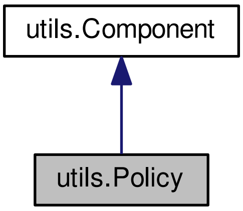
- src/utils/analysis.py

8.49 utils.Policy Class Reference

Inheritance diagram for utils.Policy:



Collaboration diagram for utils.Policy:



Public Member Functions

- def **__init__** (self, train, validation, test, cuda)
- def **train** (self)
- def **validation** (self)
- def **test** (self)
- def **cuda** (self)

Static Public Attributes

- **interface**

8.49.1 Member Data Documentation

8.49.1.1 utils.Policy.interface [static]

Initial value:

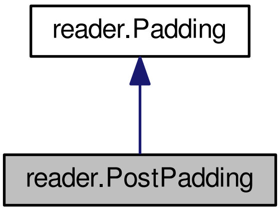
```
= Interface(**{
        'cuda':          (0, None),
        'train':         (1, None),
        'validation':    (2, None),
        'test':          (3, None)
    })
```

The documentation for this class was generated from the following file:

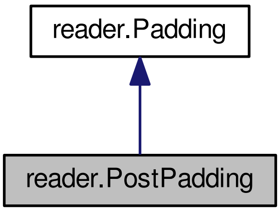
- src/utils/utils.py

8.50 reader.PostPadding Class Reference

Inheritance diagram for reader.PostPadding:



Collaboration diagram for reader.PostPadding:



Public Member Functions

- def **__init__**
- def **__call__**
- def **create_batch**

Static Public Attributes

- bool **abstract** = False

8.50.1 Detailed Description

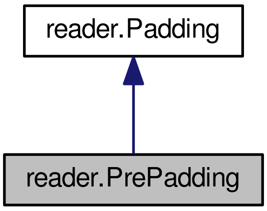
```
Data is padded during the training iterations. Padding is determined by the longest
sequence in the batch.
```

The documentation for this class was generated from the following file:

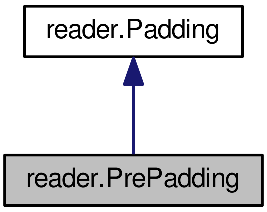
- src/utils/reader.py

8.51 reader.PrePadding Class Reference

Inheritance diagram for reader.PrePadding:



Collaboration diagram for reader.PrePadding:



Public Member Functions

- def **create_batch**
- def **__init__**
- def **__call__**

Static Public Attributes

- bool **abstract** = False

8.51.1 Detailed Description

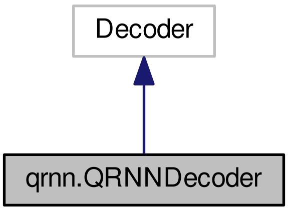
```
Data is padded previously to the training iterations. The padding is determined
by the longest sequence in the data segment.
```

The documentation for this class was generated from the following file:

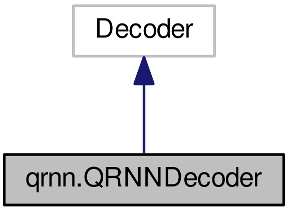
- src/utils/reader.py

8.52 qrnn.QRNNDecoder Class Reference

Inheritance diagram for qrnn.QRNNDecoder:



Collaboration diagram for qrnn.QRNNDecoder:

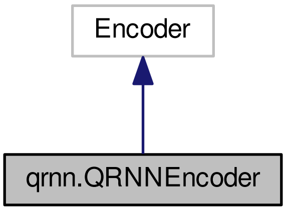


The documentation for this class was generated from the following file:

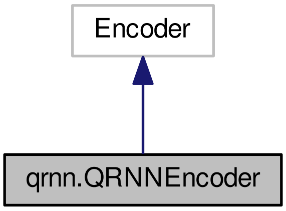
- src/components/decoders/qrnn.py

8.53 qrnn.QRNNEncoder Class Reference

Inheritance diagram for qrnn.QRNNEncoder:



Collaboration diagram for qrnn.QRNNEncoder:

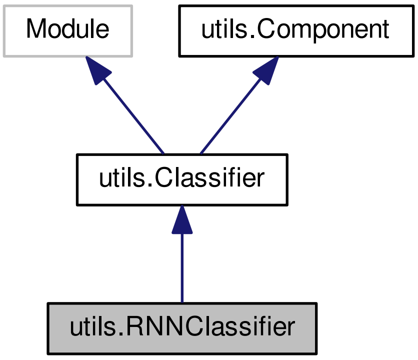


The documentation for this class was generated from the following file:

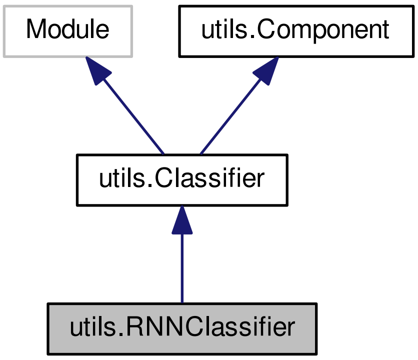
- src/components/encoders/qrnn.py

8.54 utils.RNNClassifier Class Reference

Inheritance diagram for utils.RNNClassifier:



Collaboration diagram for utils.RNNClassifier:



Public Member Functions

- def **__init__**
- def forward (self, args, inputs, lengths, kwargs)
- def **optimizer** (self)

Static Public Attributes

- bool **abstract** = False
- **interface**

8.54.1 Detailed Description

```
Recurrent discriminator module for the unsupervised neural translation task.
```

8.54.2 Member Function Documentation

8.54.2.1 def utils.RNNClassifier.forward (*self*, *args*, *inputs*, *lengths*, *kwargs*)

```
Forward step for the discriminator.

:param inputs:
    Variable, (batch_size, input_size), where input_size is equal to the encoder’s
    hidden_size.

:param lengths:

:return final_output:
    Variable, (batch_size, 1).
```

8.54.3 Member Data Documentation

8.54.3.1 utils.RNNClassifier.interface [static]

Initial value:

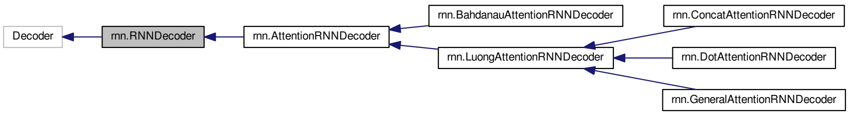
```
= Interface(**{
        **Classifier.interface.dictionary,
        ’num_layers’: (Interface.last_key(Classifier.interface.dictionary) + 1, None)
    })
```

The documentation for this class was generated from the following file:

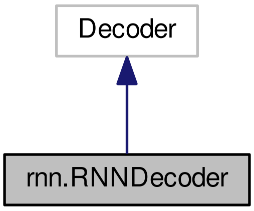
- src/components/utils/utils.py

8.55 rnn.RNNDecoder Class Reference

Inheritance diagram for rnn.RNNDecoder:


Decoder
rnn.RNNDecoder
rnn.AttentionRNNDecoder
rnn.BahdanauAttentionRNNDecoder
rnn.LuongAttentionRNNDecoder
rnn.ConcatAttentionRNNDecoder
rnn.DotAttentionRNNDecoder
rnn.GeneralAttentionRNNDecoder

Collaboration diagram for rnn.RNNDecoder:



Public Member Functions

- def **__init__**
- def init_parameters (self)
- def init_optimizer (self)
- def **forward**
- def tokens (self)
- def **tokens**
- def **output_types** (self)
- def optimizers (self)
- def state (self)
- def **state**

Public Attributes

- **embedding**
- **output_layer**

Static Public Attributes

- **interface**
- bool **abstract** = False

8.55.1 Detailed Description

```
An implementation of recurrent decoder unit for the sequence to sequence model.
```

8.55.2 Member Function Documentation

8.55.2.1 def rnn.RNNDecoder.init_optimizer (*self, Decoder*)

```
Initializes the optimizer for the decoder.
```

8.55.2.2 def rnn.RNNDecoder.init_parameters (*self, Decoder*)

Calls the parameter setter, which initializes the Parameter type attributes.
After initialization, the main components of the decoder, which require the previously initialized parameter values, are created as well.

8.55.2.3 def rnn.RNNDecoder.optimizers (*self, list*)

Property for the optimizers of the decoder.

8.55.2.4 def rnn.RNNDecoder.state (*self, dict*)

Property for the optimizers of the decoder.

8.55.2.5 def rnn.RNNDecoder.tokens (*self, dict*)

Property for the tokens of the decoder.

8.55.3 Member Data Documentation

8.55.3.1 rnn.RNNDecoder.interface [static]

Initial value:

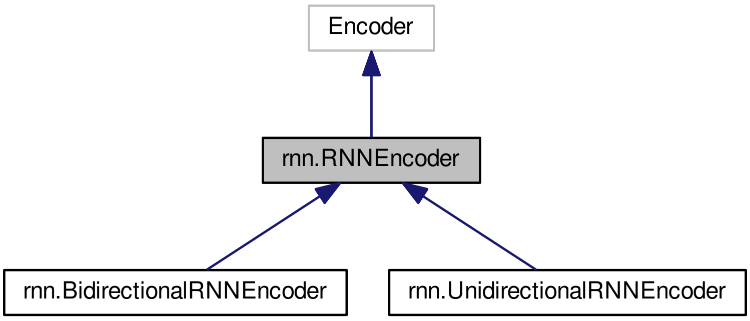
```
= Interface(**{
        ’hidden_size’:      (0, None),
        ’recurrent_type’:   (1, None),
        ’num_layers’:       (2, None),
        ’optimizer_type’:   (3, None),
        ’learning_rate’:    (4, None),
        ’max_length’:       (5, None),
        ’cuda’:             (6, ’Experiment:Policy:cuda$’),
        ’embedding_size’:   (7, ’embedding_size$’),
        ’input_size’:       (8, ’embedding_size$’)
    })
```

The documentation for this class was generated from the following file:

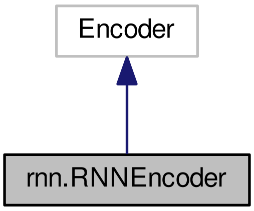
- src/components/decoders/rnn.py

8.56 rnn.RNNEncoder Class Reference

Inheritance diagram for rnn.RNNEncoder:



Collaboration diagram for rnn.RNNEncoder:



Public Member Functions

- def __init__ (self, parameter_setter)
- def **forward** (self, args, kwargs)
- def init_optimizer (self)
- def **output_types** (self)
- def optimizers (self)
- def state (self)
- def state (self, state)

Public Attributes

- **embedding**

Static Public Attributes

- **interface**
- bool **abstract** = True

8.56.1 Detailed Description

```
Recurrent encoder module of the sequence to sequence model.
```

8.56.2 Constructor & Destructor Documentation

8.56.2.1 def rnn.RNNEncoder.__init__ (*self*, *parameter_setter*)

```
A recurrent encoder module for the sequence to sequence model.
:param parameter_setter: ParameterSetter object, that requires the following parameters.
    -:parameter hidden_size: int, size of recurrent layer of the LSTM/GRU.
    -:parameter recurrent_layer: str, name of the recurrent layer ('GRU', 'LSTM').
    -:parameter embedding_size: int, dimension of the word embeddings.
    -:parameter optimizer: Optimizer, for parameter optimalization.
    -:parameter learning_rate: float, learning rate.
    -:parameter use_cuda: bool, True if the device has cuda support.
```

8.56.3 Member Function Documentation

8.56.3.1 def rnn.RNNEncoder.init_optimizer (*self*)

```
Initializes the optimizer for the encoder.
```

8.56.3.2 def rnn.RNNEncoder.optimizers (*self*)

```
Property for the optimizers of the encoder.
```

8.56.3.3 def rnn.RNNEncoder.state (*self*)

```
Property for the state of the encoder.
```

8.56.3.4 def rnn.RNNEncoder.state (*self*, *state*)

```
Setter method for the weights of the encoder, and the optimizer.
:param state: dict, containing the states.
```

8.56.4 Member Data Documentation

8.56.4.1 rnn.RNNEncoder.interface [static]

Initial value:

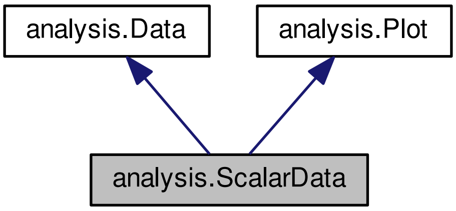
```
= Interface(**{
        'hidden_size':      (0, None),
        'recurrent_type':   (1, None),
        'num_layers':       (2, None),
        'optimizer_type':   (3, None),
        'learning_rate':    (4, None),
        'cuda':             (5, 'Experiment:Policy:cuda$'),
        'embedding_size':   (6, 'embedding_size$')
    })
```

The documentation for this class was generated from the following file:

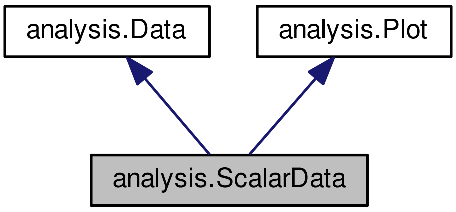
- src/components/encoders/rnn.py

8.57 analysis.ScalarData Class Reference

Inheritance diagram for analysis.ScalarData:



Collaboration diagram for analysis.ScalarData:



Public Member Functions

- def **__init__** (self)
- def **add** (self, identifier, value)
- def **average** (self, identifiers=None)
- def **get_required_keys** (self)

Static Public Member Functions

- def **summed_average** (scalar_iterable, identifiers=None)
- def **display** (data, plot_size, epochs, epoch_range, identifiers=None)

Additional Inherited Members

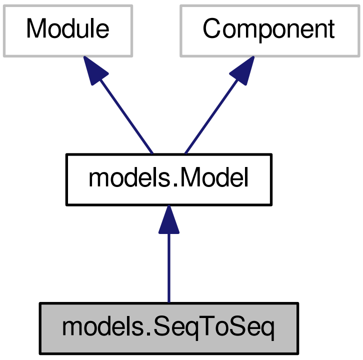
8.57.1 Detailed Description

The documentation for this class was generated from the following file:

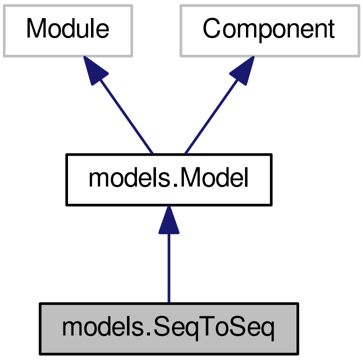
- src/utils/analysis.py

8.58 models.SeqToSeq Class Reference

Inheritance diagram for models.SeqToSeq:



Collaboration diagram for models.SeqToSeq:



Public Member Functions

- def __init__ (self, encoder, decoder)
- def forward (self, inputs, lengths, targets, max_length)
- def freeze (self)
- def unfreeze (self)
- def optimizers (self)
- def output_size (self)
- def decoder_tokens (self)
- def decoder_tokens (self, tokens)
- def output_types (self)
- def state (self)
- def state (self, state)

Public Attributes

- **encoder**
- **decoder**

Static Public Attributes

- **interface**
- bool **abstract** = False

8.58.1 Detailed Description

```
Sequence to sequence model according to the one described in:

    https://arxiv.org/abs/1409.3215

The model has two main components, an encoder and a decoder, which may be
implemented as recurrent or convolutional units. The main principle of this technique
is to map a sequence - in case of translation - a sentence to another sentence,
by encoding it to a fixed size representation, and then decoding this latent meaning
vector to the desired sequence.
```

8.58.2 Constructor & Destructor Documentation

8.58.2.1 def models.SeqToSeq.__init__ (*self, encoder, decoder*)

```
An instance of ta sequence to sequence model.

:param encoder:
    Encoder, an encoder instance.

:param decoder:
    Decoder, a decoder instance.
```

8.58.3 Member Function Documentation

8.58.3.1 def models.SeqToSeq.decoder_tokens (*self*)

```
Tokens used by the decoder, for special outputs.
```

8.58.3.2 def models.SeqToSeq.decoder_tokens (*self, tokens*)

```
Setter for the tokens, that will be used by the decoder.
```

8.58.3.3 def models.SeqToSeq.forward (*self, inputs, lengths, targets, max_length*)

```
Forward step of the sequence to sequence model.

:param inputs:
    Variable, containing the ids of the tokens for the input sequence.

:param targets:
    Variable, containing the ids of the tokens for the target sequence.

:param max_length:
    int, the maximum length of the decoded sequence.

:param lengths:
    Ndarray, containing the lengths of the original sequences.

:return outputs:
    dict, containing the concatenated outputs of the encoder and decoder.
```

8.58.3.4 def models.SeqToSeq.freeze (*self*)

8.58.3.5 def models.SeqToSeq.optimizers (*self*)

```
Convenience function for the optimizers of the encoder and decoder.
:return: dict, containing the names and instances of optimizers for the encoder/decoder
 and the currently used embeddings.
```

8.58.3.6 def models.SeqToSeq.output_size (*self*)

```
THe dimension of the decoder’s output layer.
```

8.58.3.7 def models.SeqToSeq.output_types (*self*)

8.58.3.8 def models.SeqToSeq.state (*self*)

8.58.3.9 def models.SeqToSeq.state (*self, state*)

8.58.3.10 def models.SeqToSeq.unfreeze (*self*)

8.58.4 Member Data Documentation

8.58.4.1 models.SeqToSeq.interface [static]

Initial value:

```
= Interface(**{
        ’encoder’:    (0, Encoder),
        ’decoder’:    (1, Decoder)
    })
```

The documentation for this class was generated from the following file:

- src/models/models.py

8.59 session.Session Class Reference

Public Member Functions

- def **__init__** (self, experiment, model_dir, clear=False)
- def **save_state** (self, state, name='checkpoint')
- def **test** (self)
- def **evaluate** (self)
- def **train** (self)
- def **task** (self)
- def **output_dir** (self)
- def **model_dir** (self)
- def **state** (self)

Public Attributes

- **experiment**
- **interrupted**

Static Public Attributes

- string **CHECKPOINT_DIR** = 'checkpoints'
- string **OUTPUT_DIR** = 'outputs'
- string **LOG_DIR** = 'info.log'
- string **INTERRUPT** = 'interrupt'

The documentation for this class was generated from the following file:

- src/utils/session.py

8.60 session.TestContext Class Reference

Public Member Functions

- def **__init__** (self, session)
- def **__enter__** (self)
- def **__exit__** (self, exc_type, exc_val, exc_tb)
- def **test** (self)
- def **save** (self)

Static Public Attributes

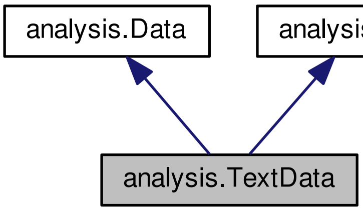
- string **TEST_FILE** = 'test.pt'

The documentation for this class was generated from the following file:

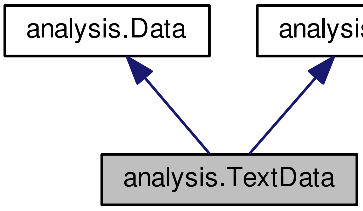
- src/utils/session.py

8.61 analysis.TextData Class Reference

Inheritance diagram for analysis.TextData:



Collaboration diagram for analysis.TextData:



Public Member Functions

- def **__init__** (self)
- def **as_str** (self, identifier)
- def **add** (self, identifier, value)
- def **get_required_keys** (self)
- def **calculate_metrics** (self)

Static Public Member Functions

- def **display** (data, plot_size, epochs, epoch_range, identifiers=None, params)

Additional Inherited Members

8.61.1 Detailed Description

The documentation for this class was generated from the following file:

- src/utils/analysis.py

8.62 session.TrainingContext Class Reference

Public Member Functions

- def **__init__** (self, session)
- def **__enter__** (self)
- def **__exit__** (self, exc_type, exc_val, exc_tb)
- def **train** (self)
- def **epochs** (self)

Public Attributes

- **epoch**

Static Public Attributes

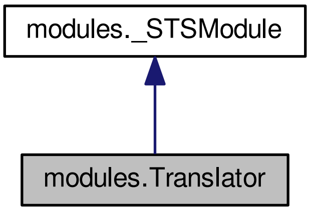
- int **EPOCHS** = 10000

The documentation for this class was generated from the following file:

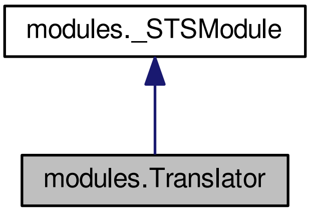
- src/utils/session.py

8.63 modules.Translator Class Reference

Inheritance diagram for modules.Translator:



Collaboration diagram for modules.Translator:



Public Member Functions

- def **__init__**
- def __call__ (self, batch, input_lang_index, target_lang_index, forced_targets=True)

8.63.1 Detailed Description

8.63.2 Member Function Documentation

8.63.2.1 def modules.Translator.__call__ (*self*, *batch*, *input_lang_index*, *target_lang_index*, *forced_targets* = True)

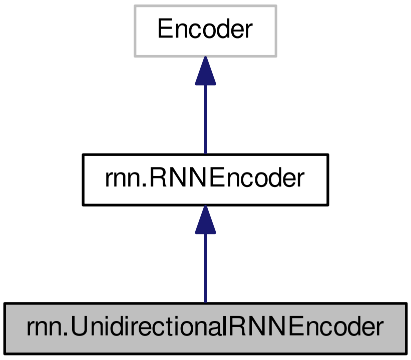
```
Implementation of a step of auto-encoding. The look up tables of the model are fitted to the
provided inputs, and the <LNG> are substituted with the appropriate token. In this case the token
is the source language token. The inputs are then transformed by a noise function, and then fed
through the model. If reguralization is applied, the encoder outputs are fetched from the output
of the model, which is used by the discriminator to apply an adversarial reguralization on these
outputs.

:param input_lang_index:
    An int value, that represents the index of the language. This value will serve as
    the index of the substitution token for the input batch.

:param target_lang_index:

:param batch:
    A list, containing the batches from the input pipelines.

:param forced_targets:

:return loss:
    A scalar loss value, indicating the average loss of the auto encoder.

:return outputs:
    A dictionary, that contains the outputs of the model. The types (keys) contained
    by this dictionary depends on the model specifications.
```

The documentation for this class was generated from the following file:

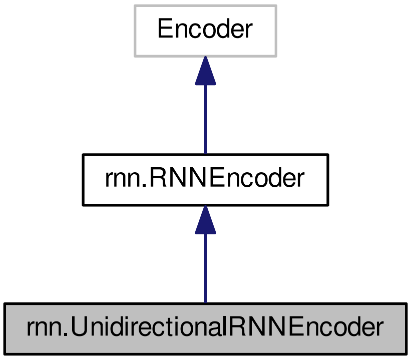
- src/modules/modules.py

8.64 rnn.UnidirectionalRNNEncoder Class Reference

Inheritance diagram for rnn.UnidirectionalRNNEncoder:



Collaboration diagram for rnn.UnidirectionalRNNEncoder:



Public Member Functions

- def **__init__** (self, parameter_setter)
- def init_parameters (self)
- def forward (self, inputs, lengths)

Static Public Attributes

- **interface** = RNNEncoder.interface
- bool **abstract** = False

Additional Inherited Members

8.64.1 Detailed Description

8.64.2 Member Function Documentation

8.64.2.1 def rnn.UnidirectionalRNNEncoder.forward (*self, inputs, lengths*)

```
A forward step of the encoder. The batch of sequences with word ids are
packed into padded_sequence object, which are processed by the recurrent layer.

:param inputs:
    Variable, (batch_size, sequence_length) containing the ids of the words.

:param lengths:
    Ndarray, containing the real lengths of the sequences in the batch (prior to padding).

:return outputs:
    Variable, (batch_size, sequence_length, vocab_size) the output at each time
    step of the encoder.

:return hidden_state:
    Variable, (num_layers * directions, batch_size, hidden_size) the final hidden state.
```

8.64.2.2 def rnn.UnidirectionalRNNEncoder.init_parameters (*self*)

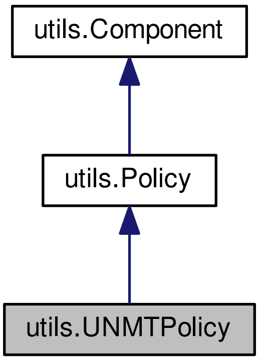
```
Calls the parameter setter, which initializes the Parameter type attributes.
After initialization, the main components of the encoder, which require the previously
initialized parameter values, are created as well.
```

The documentation for this class was generated from the following file:

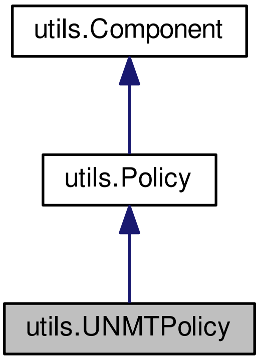
- src/components/encoders/rnn.py

8.65 utils.UNMTPolicy Class Reference

Inheritance diagram for utils.UNMTPolicy:



Collaboration diagram for utils.UNMTPolicy:



Public Member Functions

- def **__init__** (self, train, validation, test, add_language_token, cuda)

Public Attributes

- **train_tf_ratio**
- **train_noise**
- **validation_tf_ratio**
- **validation_noise**
- **test_tf_ratio**
- **test_noise**
- **add_language_token**

Static Public Attributes

- **interface**
- bool **abstract** = False

8.65.1 Member Data Documentation

8.65.1.1 utils.UNMTPolicy.interface [static]

Initial value:

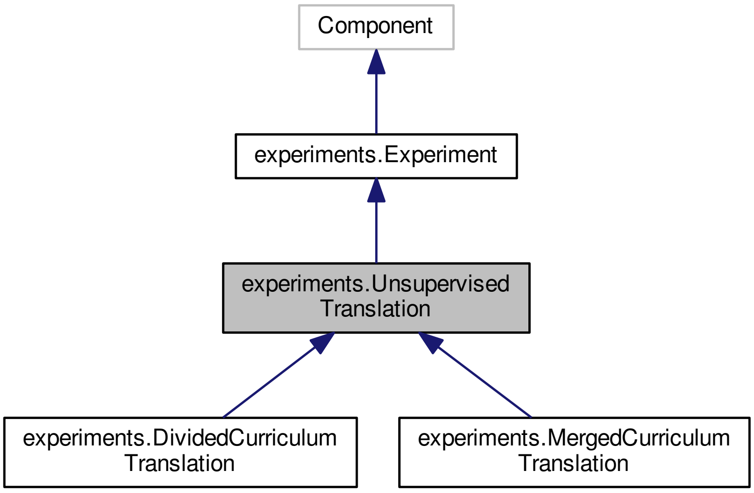
```
= Interface(**{
        **Policy.interface.dictionary,
        'add_language_token': (Interface.last_key(Policy.interface.dictionary) + 1, None),
    })
```

The documentation for this class was generated from the following file:

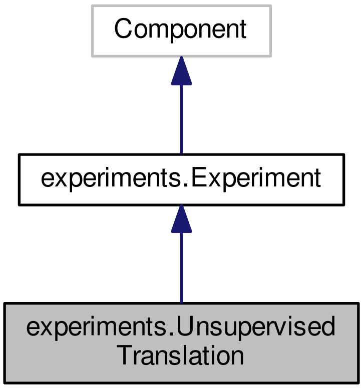
- src/utils/utils.py

8.66 experiments.UnsupervisedTranslation Class Reference

Inheritance diagram for experiments.UnsupervisedTranslation:


Component
experiments.Experiment
experiments.Unsupervised Translation
experiments.DividedCurriculum Translation
experiments.MergedCurriculum Translation

Collaboration diagram for experiments.UnsupervisedTranslation:



Public Member Functions

- def **__init__**
- def **train**
- def **validate** (self)

- def **test** (self)
- def **evaluate** (self)
- def **state** (self)
- def **state** (self, value)

Static Public Member Functions

- def **clear_optimizers**
- def **step_optimizers**
- def **freeze**
- def **unfreeze**

Public Attributes

- **reguralize**

Static Public Attributes

- **interface**
- bool **abstract** = False

8.66.1 Detailed Description

```
Translation experiment, without parallel corpus. The method follows the main principles described
in this article:

    https://arxiv.org/abs/1711.00043

The main goal of this experiment is to train a denoising auto-encoder, that learns to map sentences to
sentences in two ways. The first way is to transform a noisy version of the source sentence to it’s
original form, and the second way is to transform a translated version of a sentence to it’s original form.
There is an additional factor during training, which is an adversarial reguralization, that learns to
discriminate the hidden representations of the source and target languages.
```

8.66.2 Member Data Documentation

8.66.2.1 experiments.UnsupervisedTranslation.interface [static]

Initial value:

```
= Interface(**{
        'policy':                (0, Policy),
        'language_identifiers': (1, None),
        'languages':            (2, Language),
        'model':                (3, Model),
        'initial_translator':   (4, WordTranslator),
        'reguralizer':          (5, Classifier)
    })
```

The documentation for this class was generated from the following file:

- src/experiments/experiments.py

8.67 session.ValidationContext Class Reference

Public Member Functions

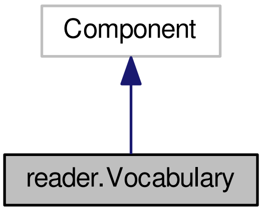
- def __**init**__ (self, session)
- def __**enter**__ (self)
- def __**exit**__ (self, exc_type, exc_val, exc_tb)
- def **validate** (self)
- def **save** (self, epoch, train_log)

The documentation for this class was generated from the following file:

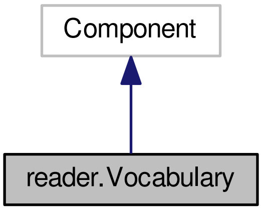
- src/utils/session.py

8.68 reader.Vocabulary Class Reference

Inheritance diagram for reader.Vocabulary:



Collaboration diagram for reader.Vocabulary:



Public Member Functions

- def **__init__**
- def __call__ (self, expression)
- def tokens (self)
- def embedding (self)
- def embedding_size (self)
- def vocab_size (self)

Public Attributes

- **requires_grad**

Static Public Attributes

- bool **abstract** = False
- **interface**

8.68.1 Detailed Description

```
Wrapper class for the lookup tables of the languages.
```

8.68.2 Member Function Documentation

8.68.2.1 def reader.Vocabulary.__call__ (*self*, *expression*)

```
Translates the given expression to it’s corresponding word or id.

:param expression:
    str or int, if str (word) is provided, then the id will be returned, and
    the behaviour is the same for the other case.

:return:
    int or str, (id or word) of the provided expression.
```

8.68.2.2 def reader.Vocabulary.embedding (*self*)

```
Property for the embedding layer.
```

8.68.2.3 def reader.Vocabulary.embedding_size (*self*)

```
Property for the dimension of the embeddings.
```

8.68.2.4 def reader.Vocabulary.tokens (*self*)

```
Property for the tokens of the language.

:return:
    dict, <UNK>, <EOS>, <PAD> and <SOS> tokens with their ids.
```

8.68.2.5 def reader.Vocabulary.vocab_size (*self*)

```
Property for the dimension of the embeddings.
```

8.68.3 Member Data Documentation

8.68.3.1 reader.Vocabulary.interface [static]

Initial value:

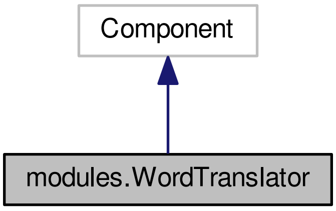
```
= Interface(**{
        'vocab_path':             (0, None),
        'provided_embeddings':    (1, None),
        'fixed_embeddings':       (2, None),
        'cuda':                   (3, 'Experiment:Policy:cuda'),
        'language_identifiers':   (4, 'Experiment:language_identifiers')
    })
```

The documentation for this class was generated from the following file:

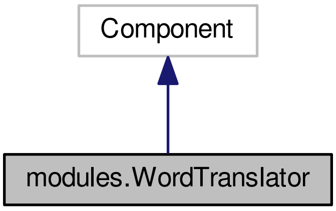
- src/utils/reader.py

8.69 modules.WordTranslator Class Reference

Inheritance diagram for modules.WordTranslator:



Collaboration diagram for modules.WordTranslator:



Public Member Functions

- def **__init__** (self, dictionaries)
- def **__call__** (self, batch, input_lang_index, target_lang_index, forced_targets=True)
- def **vocabs** (self)
- def **vocabs** (self, value)
- def **language_tokens_required** (self)
- def **language_tokens_required** (self, value)
- def **cuda** (self)
- def **cuda** (self, value)

Static Public Attributes

- bool **abstract** = False
- **interface**

8.69.1 Member Data Documentation

8.69.1.1 modules.WordTranslator.interface [static]

Initial value:

```
= Interface(**{
        'dictionaries':   (0, None)
    })
```

The documentation for this class was generated from the following file:

- src/modules/modules.py

Index