Framenet Tools

Release 0.0.1

Aug 27, 2019

Contents:

Provides functionality to find Frame Evoking Elements in raw text and predict their corresponding frames. Furthermore possible spans of roles can be found and assigned. Models can be trained either on the given files or on any annotated file in a supported format (For more information look at the section formats).

Find it on GitHub: framenet tools

CHAPTER 1

Installation

- Clone repository or download files
- Enter the directory
- Run: `pip install -e .`

1.1 Setup

- `framenet_tools download` acquires all required data and extracts it , optionally `--path` can be used to specify a custom path; default is the current directory. NOTE: After extraction the space occupied amounts up to around 9GB!
- `framenet_tools convert` can now be used to generate the CoNLL datasets This function is analogous to pyfn and simply propagates the call.
- `framenet_tools train` trains a new model on the training files and saves it, optionally `--use_eval_files` can be specified to train on the evaluation files as well. NOTE: Training can take a few minutes, depending on the hardware.

For further information run `framenet_tools --help`

1.1.1 Alternative

Alternatively conversion.sh provides a also the ability to convert FN data to CoNLL using pyfn. In this case, manually download and extract the FrameNet dataset and adjust the path inside the script.

CHAPTER 2

Usage

The following functions both require a pretrained model, which can be generated using `framenet_tools train` as explained previously.

- Stages:The System is split into 4 distinct pipeline stages, namely:
  - 1 Frameevoking element identification
  - 2 Frame identification
  - 3 Spanidentification (WIP)
  - 4 Role identification (WIP)

Each stage can individually be trained by calling it e.g. `--frameid`. Also combinations of mutliple stages are possible. This can be done for every option. NOTE: A usage of `evaluate` or `predict` requires a previous training of the same stage level!

- `framenet_tools predict --path [path]` annotates the given raw text file located at `--path` and prints the result. Optionally `--out_path` can be used to write the results directly to a file. Also a prediction can be limited to a certain stage by specifying it (e.g. `--feeid`). NOTE: As the stages build on the previous ones, this option represents a upper bound.
- `framenet_tools evaluate` evaluates the F1-Score of the model on the evaluation files. Here, evaluation can be exclusively limited to a certain stage.

2.1 Logging

Training automatically logs the loss and accuracy of the train- and devset in TensorBoard format.

- `tensorboard --logdir=runs` can beused to run TensorBoard and visualize the data.

2.2 Formats

Currently support formats include:

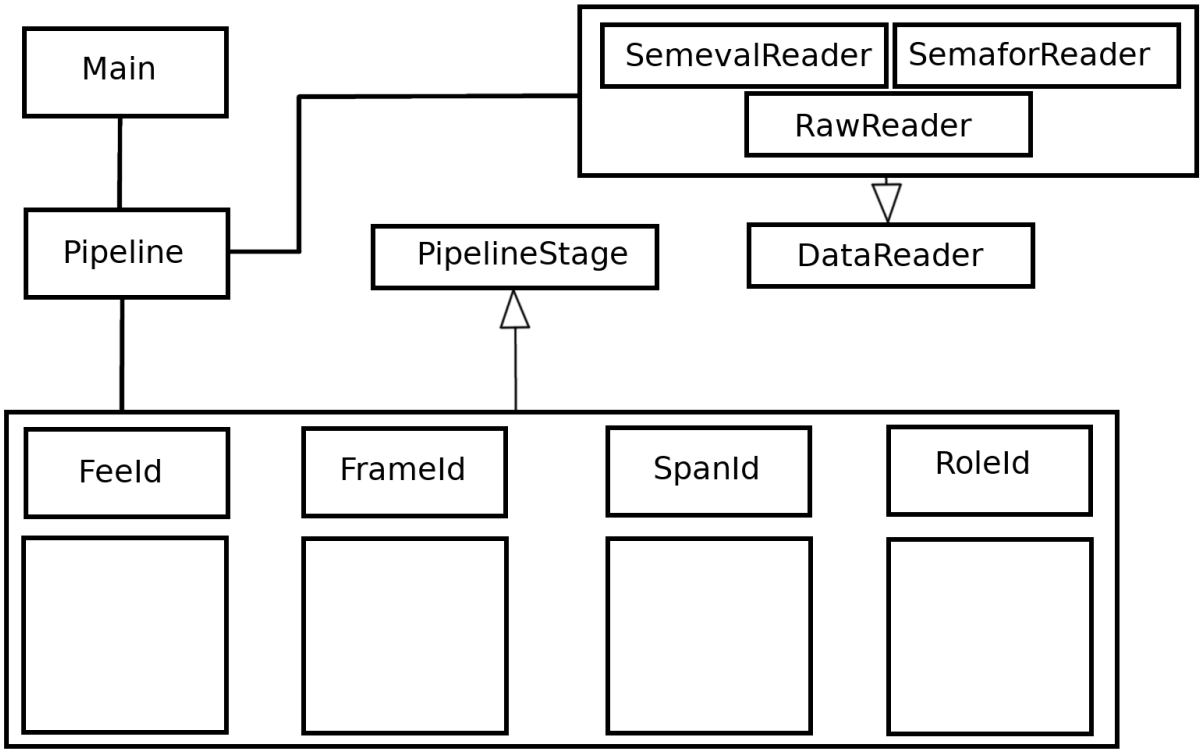
- Raw text
- SEMEVAL XML: the format of the SEMEVAL 2007 shared task 19 on frame semantic structure extraction
- SEMAFOR CoNLL: the format used by the SEMAFOR parser

NOTE: If the format is not supported, pyfn might be providing a conversion.

CHAPTER 3

Documentation

3.1 Architecture



The complete source code is available on GitHub.

3.2 Code Documentation

3.2.1 framenet_tools package

Subpackages

framenet_tools.data_handler package

Submodules

framenet_tools.data_handler.annotation module

class framenet_tools.data_handler.annotation.**Annotation**(*frame: str = 'Default'*, *fee: str = None*, *position: int = None*, *fee_raw: str = None*, *sentence: List[str] = []*, *roles: List[str] = []*, *role_positions: List[Tuple[int, int]] = []*)

Bases: object

Annotation class

Saves and manages all data of one frame for a given sentence.

create_handle()

Helper function for ease of programmatic comparison

NOTE: FEE is not compared due to possible differences during preprocessing!

Returns A handle consisting of all data saved in this object

framenet_tools.data_handler.frame_embedding_manager module

class framenet_tools.data_handler.frame_embedding_manager.**FrameEmbeddingManager**(*path: str = 'data/frame_em*

Bases: object

Loads and provides the specified frame-embeddings

embed(*frame: str*)

Converts a given frame to its embedding

Parameters **frame** – The frame to embed

Returns The embedding (n-dimensional vector)

read_frame_embeddings()

Loads the previously specified frame embedding file into a dictionary

string_to_array(*string: str*)

Helper function Converts a string of an array back into an array

NOTE: specified for float arrays !!!

Parameters **string** – The string of an array

Returns The array

framenet_tools.data_handler.rawreader module

class framenet_tools.data_handler.rawreader.**RawReader**(*cM: framenet_tools.config.ConfigManager*, *raw_path: str = None*)

Bases: *framenet_tools.data_handler.reader.DataReader*

A reader for raw text files.

Inherits from DataReader

read_raw_text(*raw_path: str = None*)

Reads a raw text file and saves the content as a dataset

NOTE: Applying this function removes the previous dataset content

Parameters **raw_path** – The path of the file to read

Returns

framenet_tools.data_handler.reader module

class framenet_tools.data_handler.reader.**DataReader**(*cM: framenet_tools.config.ConfigManager*)

Bases: object

The top-level DataReader

Stores all loaded data from every reader.

embed_frame(*frame: str*)

Embeds a single frame.

NOTE: if the embeddings of the frame can not be found, a random set of values is generated.

Parameters **frame** – The frame to embed

Returns The embedding of the frame

embed_frames(*force: bool = False*)

Embeds all the sentences that are currently loaded.

NOTE: if forced, overrides embedded data inside of the annotation objects

Parameters **force** – If true, embeddings are generate even if they already exist

Returns

embed_word(*word: str*)

Embeds a single word

Parameters **word** – The word to embed

Returns The vector of the embedding

embed_words(*force: bool = False*)

Embeds all words of all sentences that are currently saved in "sentences".

NOTE: Can erase all previously embedded data!

Parameters **force** – If true, all previously saved embeddings will be overwritten!

Returns

export_to_json (*path: str*)
Exports the list of annotations to a json file

Parameters **path** – The path of the json file

Returns

generate_pos_tags (*force: bool = False*)
Generates the POS-tags of all sentences that are currently saved.

Parameters **force** – If true, the POS-tags will overwrite previously saved tags.

Returns

get_annotations (*sentence: List[str] = None*)
Returns the annotation object for a given sentence.

Parameters **sentence** – The sentence to retrieve the annotations for.

Returns A annoation object

loaded (*is_annotated: bool*)
Helper for setting flags

Parameters **is_annotated** – flag if loaded data was annotated

Returns

framenet_tools.data_handler.semaforreader module

class framenet_tools.data_handler.semaforreader.**SemaforReader** (*cM: framenet_tools.config.ConfigManager, path_sent: str = None, path_elements: str = None*)

Bases: *framenet_tools.data_handler.reader.DataReader*

A reader for the Semafor ConLL format

Inherits from DataReader

digest_raw_data (*elements: list, sentences: list*)
Converts the raw elements and sentences into a nicely structured dataset

NOTE: This representation is meant to match the one in the "frames-files"

Parameters

- **elements** – the annotation data of the given sentences
- **sentences** – the sentences to digest

Returns

digest_role_data (*element: str*)
Parses a string of role information into the desired format

Parameters **element** – The string containing the role data

Returns A pair of two concurrent lists containing the roles and their spans

read_data(*path_sent: str = None*, *path_elements: str = None*)
Reads a the sentence and elements file and saves the content as a dataset

NOTE: Applying this function removes the previous dataset content

Parameters

- **path_sent** – The path to the sentence file
- **path_elements** – The path to the elements

Returns

framenet_tools.data_handler.semevalreader module

class framenet_tools.data_handler.semevalreader.**SemevalReader**(*cM: framenet_tools.config.ConfigManager*, *path_xml: str = None*)
Bases: *framenet_tools.data_handler.reader.DataReader*

A reader for the Semeval format.

Inherits from DataReader

digest_tree(*root: <module 'xml.etree.ElementTree' from '/home/docs/.pyenv/versions/3.7.3/lib/python3.7/xml/etree/Element*
Parses the xml-tree into a DataReader object.

Parameters **root** – The root node of the tree

Returns

read_data(*path_xml: str = None*)
Reads a xml file and parses it into the datareader format.

NOTE: Applying this function removes the previous dataset content

Parameters **path_xml** – The path of the xml file

Returns

framenet_tools.data_handler.semevalreader.**char_pos_to_sentence_pos**(*start_char: int*, *end_char: int*, *words: List[str]*)
Converts positions of char spans in a sentence into word positions.

NOTE: Returned end position is represented inclusive!

Parameters

- **start_char** – The first character of the span
- **end_char** – The last character of the span
- **words** – A list of words in a sentence

Returns The start and end position of the WORD in the sentence

framenet_tools.data_handler.word_embedding_manager module

class framenet_tools.data_handler.word_embedding_manager.**WordEmbeddingManager**(*path: str = 'data/word_embed*

Bases: object

Loads and provides the specified word-embeddings

embed(*word: str*)
Converts a given word to its embedding

Parameters **word** – The word to embed

Returns The embedding (n-dimensional vector)

read_word_embeddings()
Loads the previously specified frame embedding file into a dictionary

string_to_array(*strings: List[str]*)
Helper function Converts a string of an array back into an array

NOTE: specified for float arrays !!!

Parameters **strings** – The strings of an array

Returns The array

Module contents

framenet_tools.fee_identification package

Submodules

framenet_tools.fee_identification.feeidentifier module

class framenet_tools.fee_identification.feeidentifier.**FeeIdentifier**(*cM: framenet_tools.config.ConfigMan*

Bases: object

evaluate_acc(*dataset: List[List[str]]*)
Evaluates the accuracy of the Frame Evoking Element Identifier

NOTE: F1-Score is a better way to evaluate the Identifier, because it tends to predict too many FEEs

Parameters **dataset** – The dataset to evaluate

Returns A Triple of the count of correct elements, total elements and the accuracy

identify_targets(*sentence: list*)
Identifies targets for a given sentence

Parameters **sentence** – A list of words in a sentence

Returns A list of targets

predict_fees(*mReader: framenet_tools.data_handler.reader.DataReader*)
Predicts the Frame Evoking Elements NOTE: This drops current annotation data

Returns

predict_fees_old(*dataset: List[List[str]]*)
Predicts all FEEs for a complete datset

Parameters **dataset** – The dataset to predict

Returns A list of predictions

query(*x: List[str]*)
Query a prediction of FEEs for a given sentence

Parameters **x** – A list of words in a sentence

Returns A list of predicted FEEs

framenet_tools.fee_identification.feeidentifier.**should_include_token**(*p_data: list*)
A static syntactical prediction of possible Frame Evoking Elements

Parameters **p_data** – A list of lists containing token, pos_tag, lemma and NE

Returns A list of possible FEEs

Module contents

framenet_tools.frame_identification package

Submodules

framenet_tools.frame_identification.frameidentifier module

class framenet_tools.frame_identification.frameidentifier.**FrameIdentifier**(*cM: framenet_tools.config.Co*

Bases: object

The FrameIdentifier

Manages the neural network and dataset creation needed for training and evaluation.

evaluate(*predictions: List[<MagicMock id='139642788726656'>], xs: List[str], reader: framenet_tools.data_handler.reader.DataReader*)
Evaluates the model

NOTE: for evaluation purposes use the function evaluate_file instead

Parameters

- **predictions** – The predictions the model made on xs
- **xs** – The original fed in data
- **reader** – The reader from which xs was derived

Returns

evaluate_file(*reader: framenet_tools.data_handler.reader.DataReader, predict_fees: bool = False*)
Evaluates the model on a given file set

Parameters **reader** – The reader to evaluate on

Returns A Triple of True Positives, False Positives and False Negatives

get_iter(*reader: framenet_tools.data_handler.reader.DataReader*)
Creates an Iterator for a given DataReader object.

Parameters **reader** – The DataReader object

Returns A Iterator of the dataset

load_model(*name: str*)
Loads a model from a given file

NOTE: This drops the current model!

Parameters **name** – The path of the model to load

Returns

prepare_dataset(*xs: List[str], ys: List[str], batch_size: int = None*)
Prepares the dataset and returns a BucketIterator of the dataset

Parameters

- **batch_size** – The batch_size to which the dataset will be prepared
- **xs** – A list of sentences
- **ys** – A list of frames corresponding to the given sentences

Returns A BucketIterator of the dataset

query(*annotation: framenet_tools.data_handler.annotation.Annotation*)
A simple query for retrieving the most likely frame for a given annotation.

NOTE: require are loaded network and a annotation object which has a sentence and fee!

Parameters **annotation** – The annotation containing the sentence and the fee.

Returns

query_confidence(*annotation: framenet_tools.data_handler.annotation.Annotation*, *n: int = 5*)
A deeper query for retrieving a list of likely frames for a given annotation.

NOTE: require are loaded network and a annotation object which has a sentence and fee!

Parameters

- **annotation** – The annotation containing the sentence and the fee.
- **n** – The amount of best guesses retrieved.

Returns

save_model(*name: str*)
Saves a model as a file

Parameters **name** – The path of the model to save to

Returns

train(*reader: framenet_tools.data_handler.reader.DataReader*, *reader_dev: framenet_tools.data_handler.reader.DataReader = None*)
Trains the model on the given reader.

NOTE: If no development reader is given, autostopping will be disabled!

Parameters

- **reader** – The DataReader object which contains the training data
- **reader_dev** – The DataReader object for evaluation and auto stopping

Returns

write_predictions(*file: str*, *out_file: str*, *fee_only: bool = False*)
Prints the predictions of a given file

Parameters

- **file** – The file to predict (either a raw file or annotated file set)
- **out_file** – The filename for saving the predictions
- **fee_only** – If True, only Frame Evoking Elements are predicted, NOTE: In this case there is no need for either train or load a network

Returns

framenet_tools.frame_identification.frameidentifier.**get_dataset**(*reader: framenet_tools.data_handler.reader.Da*
Loads the dataset and combines the necessary data

Parameters **reader** – The reader that contains the dataset

Returns xs: A list of sentences appended with its FEE ys: A list of frames corresponding to the given sentences

framenet_tools.frame_identification.frameidnetwork module

class framenet_tools.frame_identification.frameidnetwork.**FrameIDNetwork**(*cM: framenet_tools.config.Confi embedding_layer: <MagicMock name='mock.Embedding' id='139642788034656'>, num_classes: int*)
Bases: object

eval_model(*dev_iter: <MagicMock name='mock.Iterator' id='139642788660448'>*)
Evaluates the model on the given dataset

UPDATE: again required and integrated for evaluating the accuracy during training. Still not recommended for final evaluation purposes.

NOTE: only works on gold FEEs, therefore deprecated use f1 evaluation instead

Parameters **dev_iter** – The dataset to evaluate on

Returns The accuracy reached on the given dataset

load_model(*path: str*)
Loads the model from a given path

Parameters **path** – The path from where to load the model

Returns

predict (*dataset_iter: <MagicMock name='mock.Iterator' id='139642787948920'>*)
Uses the model to predict all given input data

Parameters **dataset_iter** – The dataset to predict

Returns A list of predictions

query (*x: List[int]*)
Query a single sentence

Parameters **x** – A list of ints representing words according to the embedding dictionary

Returns The prediction of the frame

save_model (*path: str*)
Saves the current model at the given path

Parameters **path** – The path to save the model at

Returns

train_model (*dataset_size: int, train_iter: <MagicMock name='mock.Iterator' id='139642789142312'>, dev_iter: <MagicMock name='mock.Iterator' id='139642787977256'> = None*)
Trains the model with the given dataset Uses the model specified in net

Parameters

- **dev_iter** – The dev dataset for performance measuring
- **train_iter** – The train dataset iterator including all data for training
- **dataset_size** – The size of the dataset
- **batch_size** – The batch size to use for training

Returns

Module contents

framenet_tools.role_identification package

Submodules

framenet_tools.role_identification.roleidentifier module

class framenet_tools.role_identification.roleidentifier.**RoleIdentifier**(*cM: framenet_tools.config.Config*

Bases: object

predict_roles (*annotation: framenet_tools.data_handler.annotation.Annotation*)
Predict roles for all spans contained in the given annotation object

NOTE: Manipulates the given annotation object!

Parameters **annotation** – The annotation object to predict the roles for

Returns

Module contents

framenet_tools.span_identification package

Submodules

framenet_tools.span_identification.spanidentifier module

class framenet_tools.span_identification.spanidentifier.**SpanIdentifier**(*cM: framenet_tools.config.Config*

Bases: object

The Span Identifier for predicting possible role spans of a given sentence

Includes multiple ways of predicting: -static -using allennlp -using a bilstm

generate_BIO_tags(*annotation: framenet_tools.data_handler.annotation.Annotation*)
Generates a list of (B)egin-, (I)nside-, (O)utside- tags for a given annotation.

Parameters **annotation** – The annotation to convert

Returns A list of BIO-tags

get_dataset(*annotations: List[List[framenet_tools.data_handler.annotation.Annotation]]*)
Loads the dataset and combines the necessary data

Parameters **annotations** – A List of all annotations containing all sentences

Returns xs: A list of senctences appended with its FEE ys: A list of frames corresponding to the given sentences

get_dataset_comb(*m_reader: framenet_tools.data_handler.reader.DataReader*)
Generates sentences with their BIO-tags

Parameters **m_reader** – The DataReader to create the dataset from

Returns A pair of concurrent lists containing the sequences and their labels

load()
Loads the saved model of the span identification network

Returns

predict_spans(*m_reader: framenet_tools.data_handler.reader.DataReader*)
Predicts the spans of the currently loaded dataset. The predictions are saved in the annotations.

NOTE: All loaded spans and roles are overwritten!

Returns

prepare_dataset(*xs: List[str], ys: List[str], batch_size: int = None*)
Prepares the dataset and returns a BucketIterator of the dataset

Parameters

- **batch_size** – The batch_size to which the dataset will be prepared
- **xs** – A list of sentences
- **ys** – A list of frames corresponding to the given sentences

Returns A BucketIterator of the dataset

query(*embedded_sentence: List[float], annotation: framenet_tools.data_handler.annotation.Annotation, pos_tags: List[str], use_static: bool = True*)
Predicts a possible span set for a given sentence.

NOTE: This can be done static (only using syntax) or via an LSTM.

Parameters

- **pos_tags** – The postags of the sentence
- **embedded_sentence** – The embedded words of the sentence
- **annotation** – The annotation of the sentence to predict
- **use_static** – True uses the syntactic static version, otherwise the NN

Returns A list of possible span tuples

query_all(*annotation: framenet_tools.data_handler.annotation.Annotation*)
Returns all possible spans of a sentence. Therefore all correct spans are predicted, achieving a perfect Recall score, but close to 0 in Precision.

NOTE: This creates a power set! Meaning there will be 2^N elements returned (N: words in senctence).

Parameters **annotation** – The annotation of the sentence to predict

Returns A list of ALL possible span tuples

query_nn(*embedded_sentence: List[float], annotation: framenet_tools.data_handler.annotation.Annotation, pos_tags: List[str]*)
Predicts the possible spans using the LSTM.

NOTE: In order to use this, the network must be trained beforehand

Parameters

- **pos_tags** – The postags of the sentence
- **embedded_sentence** – The embedded words of the sentence
- **annotation** – The annotation of the sentence to predict

Returns A list of possible span tuples

query_static(*annotation: framenet_tools.data_handler.annotation.Annotation*)
Predicts the set of possible spans just by the use of the static syntax tree.

NOTE: deprecated!

Parameters **annotation** – The annotation of the sentence to predict

Returns A list of possible span tuples

to_one_hot(*l: List[int]*)
Helper Function that converts a list of numerals into a list of one-hot encoded vectors

Parameters **l** – The list to convert

Returns A list of one-hot vectors

train(*mReader*, *mReaderDev*)
Trains the model on all of the given annotations.

Parameters **annotations** – A list of all annotations to train the model from

Returns

traverse_syntax_tree(*node: <MagicMock name='mock.Token' id='139642788622800'>*)
Traverses a list, starting from a given node and returns all spans of all its subtrees.

NOTE: Recursive

Parameters **node** – The node to start from

Returns A list of spans of all subtrees

framenet_tools.span_identification.spanidnetwork module

class framenet_tools.span_identification.spanidnetwork.**SpanIdNetwork**(*cM: framenet_tools.config.ConfigMa num_classes: int*)

Bases: object

eval_dev(*xs: List[<MagicMock id='139642787751976'>] = None*, *ys: List[List[int]] = None*)
Evaluates the model directly on the a prepared dataset

Parameters

- **xs** – The development sequences, given as a list of tensors
- **ys** – The labels of the sequence

Returns

load_model(*path: str*)
Loads the model from a given path

Parameters **path** – The path from where to load the model

Returns

predict(*sent: List[int]*)
Predicts the BIO-Tags of a given sentence.

Parameters **sent** – The sentence to predict (already converted by the vocab)

Returns A list of possibilities for each word for each tag

reset_hidden()
Resets the hidden states of the LSTM.

Returns

save_model(*path: str*)
Saves the current model at the given path

Parameters **path** – The path to save the model at

Returns

train_model(*xs: List[<MagicMock id='139642787396296'>], ys: List[List[int]], dev_xs: List[<MagicMock id='139642787331656'>] = None, dev_ys: List[List[int]] = None*)
Trains the model with the given dataset Uses the model specified in net

Parameters

- **xs** – The training sequences, given as a list of tensors
- **ys** – The labels of the sequences
- **dev_xs** – The development sequences, given as a list of tensors

- **dev_ys** – The labels of the sequences

Returns

Module contents

framenet_tools.stages package

Submodules

framenet_tools.stages.feeID module

class framenet_tools.stages.feeID.**FeeID**(*cM: framenet_tools.config.ConfigManager*)
Bases: *framenet_tools.pipelinestage.PipelineStage*

The Frame evoking element identification stage

Only relies on static predictions

predict(*m_reader: framenet_tools.data_handler.reader.DataReader*)
Predict the given data

NOTE: Changes the object itself

Parameters **m_reader** – The DataReader object

Returns

train(*m_reader: framenet_tools.data_handler.reader.DataReader*, *m_reader_dev: framenet_tools.data_handler.reader.DataReader*)
No training needed

Parameters

- **m_reader** – The DataReader object which contains the training data
- **m_reader_dev** – The DataReader object for evaluation and auto stopping (NOTE: not necessarily given, as the focus might lie on maximizing the training data)

Returns

framenet_tools.stages.frameID module

class framenet_tools.stages.frameID.**FrameID**(*cM: framenet_tools.config.ConfigManager*)
Bases: *framenet_tools.pipelinestage.PipelineStage*

The Frame Identification stage

predict(*m_reader: framenet_tools.data_handler.reader.DataReader*)
Predict the given data

NOTE: Changes the object itself

Parameters **m_reader** – The DataReader object

Returns

train(*m_reader: framenet_tools.data_handler.reader.DataReader*, *m_reader_dev: framenet_tools.data_handler.reader.DataReader*)
Train the frame identification stage on the given data

NOTE: May overwrite a previously saved model!

Parameters

- **m_reader** – The DataReader object which contains the training data
- **m_reader_dev** – The DataReader object for evaluation and auto stopping (NOTE: not necessarily given, as the focus might lie on maximizing the training data)

Returns

framenet_tools.stages.roleID module

class framenet_tools.stages.roleID.**RoleID**(*cM: framenet_tools.config.ConfigManager*)
Bases: *framenet_tools.pipelinestage.PipelineStage*

The Role Identification stage

predict(*m_reader: framenet_tools.data_handler.reader.DataReader*)

Parameters **m_reader** –

Returns

train(*m_reader: framenet_tools.data_handler.reader.DataReader*, *m_reader_dev: framenet_tools.data_handler.reader.DataReader*)
Trains the role identification stage

Parameters

- **m_reader** – The DataReader object which contains the training data
- **m_reader_dev** – The DataReader object for evaluation and auto stopping (NOTE: not necessarily given, as the focus might lie on maximizing the training data)

Returns

framenet_tools.stages.spanID module

class framenet_tools.stages.spanID.**SpanID**(*cM: framenet_tools.config.ConfigManager*)
Bases: *framenet_tools.pipelinestage.PipelineStage*

The Span Identification stage

predict(*m_reader: framenet_tools.data_handler.reader.DataReader*)

Parameters **m_reader** –

Returns

train(*m_reader: framenet_tools.data_handler.reader.DataReader*, *m_reader_dev: framenet_tools.data_handler.reader.DataReader*)
Train the stage on the given data

Parameters

- **m_reader** – The DataReader object which contains the training data
- **m_reader_dev** – The DataReader object for evaluation and auto stopping (NOTE: not necessarily given, as the focus might lie on maximizing the training data)

Returns

Module contents

framenet_tools.utils package

Submodules

framenet_tools.utils.postagger module

class framenet_tools.utils.postagger.**PosTagger**(*use_spacy: bool*)
Bases: object

PosTagger provides options for assigning POS-tags to sentences.

Either by spacy or nltk.

get_tags(*sentence: List[str]*)
Returns the POS-tags of a given sentence.

Parameters **sentence** – The sentence, given as a list of words

Returns A list of POS-tags

get_tags_nltk(*tokens: List[str]*)
Gets lemma, pos and NE for each token

Parameters **tokens** – A list of tokens from a sentence

Returns A 2d-Array containing lemma, pos and NE for each token

get_tags_spacy(*tokens: List[str]*)
The spacy version of the get_tags method

:param tokens:The sentence, given as a list of words :return: A list of POS-tags

framenet_tools.utils.postagger.**get_pos_constants**(*tag: str*)
Static function for tag conversion

Parameters **tag** – The given pos tag

Returns The corresponding letter

framenet_tools.utils.static_utils module

framenet_tools.utils.static_utils.**download**(*url: str*)
Downloads and extracts a file given as a url.

NOTE: The paths should NOT be changed in order for pyfn to work NOTE: Only extracts 7z files

Parameters **url** – The url from where to get the file

Returns

framenet_tools.utils.static_utils.**download_file**(*url: str*, *file_path: str*)
Downloads a file and saves at a given path

Parameters

- **url** – The URL of the file to download
- **file_path** – The destination of the file

Returns

framenet_tools.utils.static_utils.**download_frame_embeddings**()
Checks if the needed frame embeddings are already downloaded, if not they are downloaded.

Returns

framenet_tools.utils.static_utils.**download_resources**()
Checks if the required resources from nltk are installed, if not they are downloaded.

Returns

framenet_tools.utils.static_utils.**extract7z**(*path: str*)
Extracts 7z Archive

Parameters **path** – The path of the archive

Returns

framenet_tools.utils.static_utils.**extract_file**(*file_path: str*)
Extracts a zipped file

Parameters **file_path** – The file to extract

Returns

framenet_tools.utils.static_utils.**get_sentences**(*raw: str*, *use_spacy: bool = False*)
Parses a raw string of text into structured sentences. This is either done via nltk or spacy; default being nltk.

Parameters

- **raw** – A raw string of text
- **use_spacy** – True to use spacy, otherwise nltk

Returns A list of sentences, consisting of tokens

framenet_tools.utils.static_utils.**get_sentences_nltk**(*raw: str*)
The nltk version of the get_sentences method.

Parameters **raw** – A raw string of text

Returns A list of sentences, consisting of tokens

framenet_tools.utils.static_utils.**get_sentences_spacy**(*raw: str*)
The spacy version of the get_sentences method.

Parameters **raw** – A raw string of text

Returns A list of sentences, consisting of tokens

framenet_tools.utils.static_utils.**get_spacy_en_model**()
Installs the required en_core_web_sm model

NOTE: Solution for Windows? TODO :return:

framenet_tools.utils.static_utils.**load_pkl_from_path**(*str_path_file: str*)
Taken from: https://public.ukp.informatik.tu-darmstadt.de/repl4nlp17-frameEmbeddings/reader.py

Parameters **str_path_file** – The path of the pickle file to load the dict from

Returns The loaded dict

framenet_tools.utils.static_utils.**pos_to_int**(*pos: str*)
Converts a pos tag to an integer according to the static dictionary.

Parameters **pos** – The pos tag

Returns The index of the pos tag

framenet_tools.utils.static_utils.**print_dict_to_txt**(*str_path_file: str*, *dict_to_print: dict*)
Taken from: https://public.ukp.informatik.tu-darmstadt.de/repl4nlp17-frameEmbeddings/reader.py

Parameters

- **str_path_file** – The path of the dict to save to
- **dict_to_print** – The dict to save

Returns

framenet_tools.utils.static_utils.**shuffle_concurrent_lists**(*l: List[List[object]]*)
Shuffles multiple concurrent lists so that pairs of (x, y) from different lists are still at the same index.

Parameters **l** – A list of concurrent lists

Returns The list of shuffled concurrent lists

Module contents

Submodules

framenet_tools.config module

class framenet_tools.config.**ConfigManager**(*path: str = None*)
Bases: object

create_config(*path: str*)
Creates a config file and saves all necessary variables

Returns

load_config(*path: str = None*)
Loads the config file and saves all found variables

NOTE: If no config file was found, the default configs will be loaded instead

Returns A boolean - True if the config file was loaded, False if defaults were loaded

load_defaults()
Loads the builtin defaults

Returns

paths_to_string(*files: List[List[str]]*)
Helper function for turning a list of file paths into a structured string

Parameters **files** – A list of files

Returns The string containing all files

framenet_tools.evaluator module

framenet_tools.evaluator.**calc_f**(*tp: int*, *fp: int*, *fn: int*)
Calculates the F1-Score

NOTE: This follows standard evaluation metrics TAKEN FROM: Open-SESAME (https://github.com/swabhs/open-sesame)

Parameters

- **tp** – True Postivies Count
- **fp** – False Postivies Count
- **fn** – False Negatives Count

Returns A Triple of Precision, Recall and F1-Score

framenet_tools.evaluator.**evaluate_fee_identification**(*m_reader: framenet_tools.data_handler.reader.DataReader*, *original_reader: framenet_tools.data_handler.reader.DataReader*)

Evaluates the Frame Evoking Element Identification only

Parameters

- **m_reader** – The reader containing the predicted annotations
- **original_reader** – The original reader containing the gold annotations

Returns A Triple of True positives, False positives and False negatives

framenet_tools.evaluator.**evaluate_frame_identification**(*m_reader: framenet_tools.data_handler.reader.DataReader*, *original_reader: framenet_tools.data_handler.reader.DataReader*)

Evaluates the Frame Identification

Parameters

- **m_reader** – The reader containing the predicted annotations
- **original_reader** – The original reader containing the gold annotations

Returns A Triple of True positives, False positives and False negatives

framenet_tools.evaluator.**evaluate_span_identification**(*cM: framenet_tools.config.ConfigManager*, *span_identifier: framenet_tools.span_identification.spanidentifier.Span = None*)

Evaluates the span identification for its F1 score

Parameters

- **cM** – The ConfigManager containing the evaluation files
- **span_identifier** – Optionally an instance of a SpanIdentifier

Returns A Triple of Precision, Recall and F1-Score

framenet_tools.evaluator.**evaluate_stages**(*m_reader: framenet_tools.data_handler.reader.DataReader, original_reader: framenet_tools.data_handler.reader.DataReader, levels: List[int]*)

Evaluates the stages specified in levels

Parameters

- **m_reader** – The reader including the predicted data
- **original_reader** – The reader which holds the gold data
- **levels** – The levels to evaluate for

Returns A triple of Precision, Recall and the F1-Score

framenet_tools.main module

framenet_tools.main.**check_files**(*path*)

framenet_tools.main.**create_argparser**()
Creates the ArgumentParser and defines all of its arguments.

Returns the set up ArgumentParser

framenet_tools.main.**eval_args**(*parser: <MagicMock id='139642787307984'>*, *args: List[str] = None*)
Evaluates the given arguments and runs to program accordingly.

Parameters

- **parser** – The ArgumentParser for getting the specified arguments
- **args** – Possibility for manually passing arguments.

Returns

framenet_tools.main.**main**()
The main entry point

Returns

framenet_tools.pipeline module

class framenet_tools.pipeline.**Pipeline**(*cM: framenet_tools.config.ConfigManager, levels: List[int]*)
Bases: object

The SRL pipeline

Contains the stages of Frame evoking element identification, Frame identification, Span identification and Role identification.

evaluate()
Evaluates all the specified stages of the pipeline.

NOTE: Depending on the certain levels of the pipeline, the propagated error can be large!

Returns

load_dataset(*files: List[str]*)
Helper function for loading datasets.

Parameters **files** – A List of files to load the datasets from.

Returns A reader object containing the loaded data.

predict(*file: str*, *out_path: str*)
Predicts a raw file and exports the predictions to the given file. Also only predicts up to the specified level.

NOTE: Prediction is only possible up to the level on which the pipeline was trained!

Parameters

- **file** – The raw input text file
- **out_path** – The path to save the outputs to (can be None)

Returns

train(*data: List[str], dev_data: List[str] = None*)
Trains all stages up to the specified level

Parameters

- **data** – The data to train on
- **dev_data** – The data to check evaluation on

Returns

framenet_tools.pipeline.**get_stages**(*i: int*, *cM: framenet_tools.config.ConfigManager*)
Creates a list of stages up to the bound specified

Parameters **i** – The upper bound of the pipeline stages

Returns A list of stages

framenet_tools.pipelinestage module

class framenet_tools.pipelinestage.**PipelineStage**(*cM: framenet_tools.config.ConfigManager*)
Bases: abc.ABC

Abstract stage of the pipeline

predict(*m_reader: framenet_tools.data_handler.reader.DataReader*)
Predict the given data

NOTE: Changes the object itself

Parameters **m_reader** – The DataReader object

Returns

train(*m_reader: framenet_tools.data_handler.reader.DataReader*, *m_reader_dev: framenet_tools.data_handler.reader.DataReader*)
Train the stage on the given data

Parameters

- **m_reader** – The DataReader object which contains the training data
- **m_reader_dev** – The DataReader object for evaluation and auto stopping (NOTE: not necessarily given, as the focus might lie on maximizing the training data)

Returns

Module contents

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index

F

G

S

T

W