

---

# Framenet Tools

*Release 0.0.1*

Sep 23, 2022



---

## Contents:

---

<b>1</b>	<b>Installation</b>	<b>3</b>
1.1	Setup . . . . .	3
<b>2</b>	<b>Usage</b>	<b>5</b>
2.1	Logging . . . . .	5
2.2	Formats . . . . .	6
<b>3</b>	<b>Documentation</b>	<b>7</b>
3.1	Architecture . . . . .	7
3.2	Code Documentation . . . . .	8
<b>4</b>	<b>Indices and tables</b>	<b>29</b>
	<b>Python Module Index</b>	<b>31</b>
	<b>Index</b>	<b>33</b>



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](#)



- 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.





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 multiple 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. `--feedid`). NOTE: As the stages build on the previous ones, this option represents an 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 be used to run TensorBoard and visualize the data.

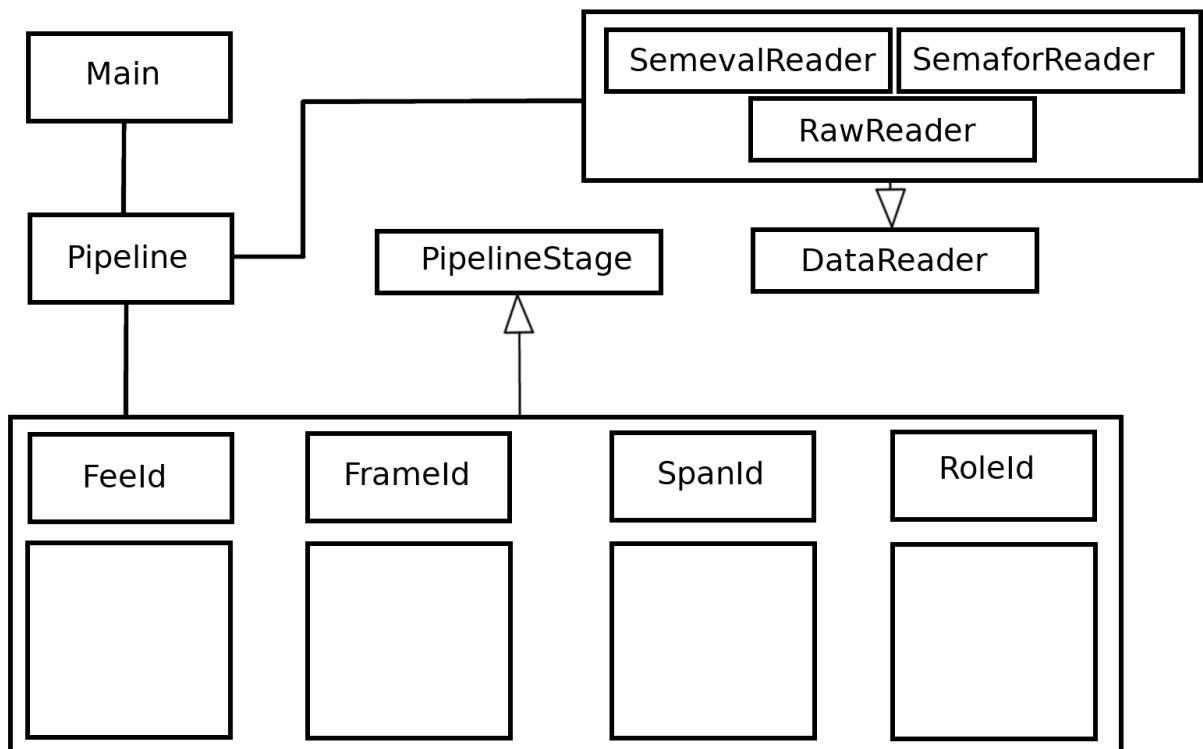
## 2.2 Formats

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.

### 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_embeddings')
```

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

**import\_from\_json** (*path: str*)

Reads the data from a given json file

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

### Returns

**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/3.7.9/lib/python3.7/xml/etree/ElementTree.py'>*)

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_embeddings')
```

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.ConfigManager)
```

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 dataset

**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.C*)

Bases: `object`

The FrameIdentifier

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

**evaluate** (*predictions: List[<MagicMock id='140524249644304'>], 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*  
*em-*  
*bed-*  
*ding\_layer:*  
*<Mag-*  
*ic-*  
*Mock*  
*name='mock.Embedding'*  
*id='140524249431824'>,*  
*num\_classes:*  
*int)*

Bases: `object`

**eval\_model** (*dev\_iter: <MagicMock name='mock.Iterator' id='140524249565456'>*)

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='140524249531024'>*)

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='140524249470416'>*, *dev\_iter*: *<MagicMock name='mock.Iterator' id='140524250806672'>* = *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

**dep\_to\_int** (*dep: str*)

Converts a dependency feature into a number

**Parameters dep** – The feature

**Returns** A consistent number

**gen\_embedding\_layer** (*reader: framenet\_tools.data\_handler.reader.DataReader*)

**Parameters reader** –

**Returns**

**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 sentences 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**

**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**

**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 sentence).

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

**Returns** A list of ALL possible span tuples

**query\_nn** (*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
- **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

**save\_model** (*name: str*)

Saves a model as a file

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

**Returns**

**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='140524249095440'>*)

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.spanidentifier.get_dataset` (*reader:*

*framenet\_tools.data\_handler.reader.DataK*

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.span\_identification.spanidnetwork module

**class** `framenet_tools.span_identification.spanidnetwork.SpanIdNetwork` (*cM:*

*framenet\_tools.config.ConfigMa*

*num\_classes:*

*int,*

*embed-*

*ding\_layer:*

*<Mag-*

*icMock*

*name='mock.Embedding'*

*id='140524249341328'>)*

Bases: `object`

**eval\_dev** (*xs: List[<MagicMock id='140524248946832'>] = 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='140524248846672'>], ys: List[List[int]], dev\_xs: List[<MagicMock id='140524248899088'>] = 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*)  
 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

## framenet\_tools.utils package

## Submodules

## framenet\_tools.utils.postagger module

**class** `framenet_tools.utils.postagger.PostTagger` (*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(m_reader:
                                                    framenet_tools.data_handler.reader.DataReader,
                                                    original_reader:
                                                    framenet_tools.data_handler.reader.DataReader)
```

Evaluates the span identification for its F1 score

**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_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='140524249116816'>, 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.pipelinstage module

**class** framenet\_tools.pipelinstage.**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`



### f

framenet\_tools, 28

framenet\_tools.config, 25

framenet\_tools.data\_handler, 12

framenet\_tools.data\_handler.annotation, 8

framenet\_tools.data\_handler.frame\_embedding\_manager, 8

framenet\_tools.data\_handler.rawreader, 9

framenet\_tools.data\_handler.reader, 9

framenet\_tools.data\_handler.semaforreader, 10

framenet\_tools.data\_handler.semevalreader, 11

framenet\_tools.data\_handler.word\_embedding\_manager, 12

framenet\_tools.evaluator, 25

framenet\_tools.fee\_identification, 13

framenet\_tools.fee\_identification.feeidentifier, 12

framenet\_tools.frame\_identification, 16

framenet\_tools.frame\_identification.frameidentifier, 13

framenet\_tools.frame\_identification.frameidnetwork, 15

framenet\_tools.main, 26

framenet\_tools.pipeline, 27

framenet\_tools.pipelinstage, 28

framenet\_tools.role\_identification, 17

framenet\_tools.role\_identification.roleidentifier, 16

framenet\_tools.span\_identification, 20

framenet\_tools.span\_identification.spanidentifier, 17

framenet\_tools.span\_identification.spanidnetwork, 19

framenet\_tools.stages, 22

framenet\_tools.stages.feeID, 20

framenet\_tools.stages.frameID, 21

framenet\_tools.stages.roleID, 21

framenet\_tools.stages.spanID, 22

framenet\_tools.utils, 25

framenet\_tools.utils.postagger, 22

framenet\_tools.utils.static\_utils, 23



## A

Annotation (class  
    *framenet\_tools.data\_handler.annotation*),  
    8

## C

calc\_f() (in module *framenet\_tools.evaluator*), 25  
char\_pos\_to\_sentence\_pos() (in module  
    *framenet\_tools.data\_handler.semevalreader*),  
    11  
check\_files() (in module *framenet\_tools.main*), 26  
ConfigManager (class in *framenet\_tools.config*), 25  
create\_argparser() (in module  
    *framenet\_tools.main*), 26  
create\_config() (*framenet\_tools.config.ConfigManager*  
    method), 25  
create\_handle() (*framenet\_tools.data\_handler.annotation.Annotation*  
    method), 8

## D

DataReader (class  
    *framenet\_tools.data\_handler.reader*), 9  
dep\_to\_int() (*framenet\_tools.span\_identification.spanidentifier.SpanIdentifier*  
    method), 17  
digest\_raw\_data()  
    (*framenet\_tools.data\_handler.semaforreader.SemaforReader*  
    method), 10  
digest\_role\_data()  
    (*framenet\_tools.data\_handler.semaforreader.SemaforReader*  
    method), 10  
digest\_tree() (*framenet\_tools.data\_handler.semevalreader.SemevalReader*  
    method), 11  
download() (in module  
    *framenet\_tools.utils.static\_utils*), 23  
download\_file() (in module  
    *framenet\_tools.utils.static\_utils*), 23  
download\_frame\_embeddings() (in module  
    *framenet\_tools.utils.static\_utils*), 23  
download\_resources() (in module  
    *framenet\_tools.utils.static\_utils*), 23

## E

embed() (*framenet\_tools.data\_handler.frame\_embedding\_manager.FrameEmbeddingManager*  
    method), 8  
embed() (*framenet\_tools.data\_handler.word\_embedding\_manager.WordEmbeddingManager*  
    method), 12  
embed\_frame() (*framenet\_tools.data\_handler.reader.DataReader*  
    method), 9  
embed\_frames() (*framenet\_tools.data\_handler.reader.DataReader*  
    method), 9  
embed\_word() (*framenet\_tools.data\_handler.reader.DataReader*  
    method), 9  
embed\_words() (*framenet\_tools.data\_handler.reader.DataReader*  
    method), 9  
eval\_args() (in module *framenet\_tools.main*), 26  
eval\_dev() (*framenet\_tools.span\_identification.spanidnetwork.SpanIdNetwork*  
    method), 19  
eval\_model() (*framenet\_tools.frame\_identification.frameidnetwork.FrameIdNetwork*  
    method), 15  
evaluate() (*framenet\_tools.frame\_identification.frameidentifier.FrameIdentifier*  
    method), 13  
evaluate() (in module *framenet\_tools.pipeline.Pipeline*  
    method), 27  
evaluate\_acc() (*framenet\_tools.fee\_identification.feeidentifier.FeeIdentifier*  
    method), 12  
evaluate\_fee\_identification() (in module  
    *framenet\_tools.evaluator*), 25  
evaluate\_file() (*framenet\_tools.frame\_identification.frameidentifier.FrameIdentifier*  
    method), 13  
evaluate\_frame\_identification() (in module  
    *framenet\_tools.evaluator*), 26  
evaluate\_span\_identification() (in module  
    *framenet\_tools.evaluator*), 26  
evaluate\_stages() (in module  
    *framenet\_tools.evaluator*), 26  
export\_to\_json() (*framenet\_tools.data\_handler.reader.DataReader*  
    method), 10  
extract7z() (in module  
    *framenet\_tools.utils.static\_utils*), 23  
extract\_file() (in module

*framenet\_tools.utils.static\_utils*), 23

## F

FeeID (class in *framenet\_tools.stages.feeID*), 20

FeeIdentifier (class in *framenet\_tools.fee\_identification.feeidentifier*), 12

FrameEmbeddingManager (class in *framenet\_tools.data\_handler.frame\_embedding\_manager*), 8

FrameID (class in *framenet\_tools.stages.frameID*), 21

FrameIdentifier (class in *framenet\_tools.frame\_identification.frameidentifier*), 13

FrameIDNetwork (class in *framenet\_tools.frame\_identification.frameidnetwork*), 15

*framenet\_tools* (module), 28

*framenet\_tools.config* (module), 25

*framenet\_tools.data\_handler* (module), 12

*framenet\_tools.data\_handler.annotation* (module), 8

*framenet\_tools.data\_handler.frame\_embedding\_manager* (module), 8

*framenet\_tools.data\_handler.rawreader* (module), 9

*framenet\_tools.data\_handler.reader* (module), 9

*framenet\_tools.data\_handler.semaforreader* (module), 10

*framenet\_tools.data\_handler.semevalreader* (module), 11

*framenet\_tools.data\_handler.word\_embedding\_manager* (module), 12

*framenet\_tools.evaluator* (module), 25

*framenet\_tools.fee\_identification* (module), 13

*framenet\_tools.fee\_identification.feeidentifier* (module), 12

*framenet\_tools.frame\_identification* (module), 16

*framenet\_tools.frame\_identification.frameembeddingmanager* (module), 13

*framenet\_tools.frame\_identification.frameidnetwork* (module), 15

*framenet\_tools.main* (module), 26

*framenet\_tools.pipeline* (module), 27

*framenet\_tools.pipelinestage* (module), 28

*framenet\_tools.role\_identification* (module), 17

*framenet\_tools.role\_identification.roleidentifier* (module), 16

*framenet\_tools.span\_identification* (module), 20

*framenet\_tools.span\_identification.spanidentifier* (module), 17

*framenet\_tools.span\_identification.spanidnetwork* (module), 19

*framenet\_tools.stages* (module), 22

*framenet\_tools.stages.feeID* (module), 20

*framenet\_tools.stages.frameID* (module), 21

*framenet\_tools.stages.roleID* (module), 21

*framenet\_tools.stages.spanID* (module), 22

*framenet\_tools.utils* (module), 25

*framenet\_tools.utils.postagger* (module), 22

*framenet\_tools.utils.static\_utils* (module), 23

## G

*gen\_embedding\_layer* () (in *framenet\_tools.span\_identification.spanidentifier.SpanIdentifier* method), 17

*generate\_BIO\_tags* () (in *framenet\_tools.span\_identification.spanidentifier.SpanIdentifier* method), 17

*generate\_hops\_tags* () (in *framenet\_tools.data\_handler.reader.DataReader* method), 10

*get\_annotations* () (in *framenet\_tools.data\_handler.reader.DataReader* method), 10

*get\_dataset* () (in *framenet\_tools.span\_identification.spanidentifier.SpanIdentifier* method), 17

*get\_dataset* () (in *framenet\_tools.frame\_identification.frameidentifier*), 15

*get\_dataset* () (in *framenet\_tools.span\_identification.spanidentifier*), 19

*get\_dataset\_comb* () (in *framenet\_tools.span\_identification.spanidentifier.SpanIdentifier* method), 17

*get\_iter* () (in *framenet\_tools.frame\_identification.frameidentifier.FrameIdentifier* method), 14

*get\_pos\_constants* () (in *framenet\_tools.utils.postagger*), 23

*get\_networks* () (in *framenet\_tools.utils.static\_utils*), 24

*get\_sentences\_nltk* () (in *framenet\_tools.utils.static\_utils*), 24

*get\_sentences\_spacy* () (in *framenet\_tools.utils.static\_utils*), 24

*get\_spacy\_en\_model* () (in *framenet\_tools.utils.static\_utils*), 24

*get\_stages* () (in *framenet\_tools.pipeline*), 27

[get\\_tags\(\)](#) ([framenet\\_tools.utils.postagger.PosTagger](#) [method](#)), [22](#)  
[get\\_tags\\_nltk\(\)](#) ([framenet\\_tools.utils.postagger.PosTagger](#) [method](#)), [23](#)  
[get\\_tags\\_spacy\(\)](#) ([framenet\\_tools.utils.postagger.PosTagger](#) [method](#)), [23](#)  
**I**  
[identify\\_targets\(\)](#) ([framenet\\_tools.fee\\_identification.feeidentifier.FeeIdentifier](#) [method](#)), [12](#)  
[import\\_from\\_json\(\)](#) ([framenet\\_tools.data\\_handler.reader.DataReader](#) [method](#)), [10](#)  
**L**  
[load\(\)](#) ([framenet\\_tools.span\\_identification.spanidentifier.SpanIdentifier](#) [method](#)), [17](#)  
[load\\_config\(\)](#) ([framenet\\_tools.config.ConfigManager](#) [method](#)), [25](#)  
[load\\_dataset\(\)](#) ([framenet\\_tools.pipeline.Pipeline](#) [method](#)), [27](#)  
[load\\_defaults\(\)](#) ([framenet\\_tools.config.ConfigManager](#) [method](#)), [25](#)  
[load\\_model\(\)](#) ([framenet\\_tools.frame\\_identification.frameidentifier.FrameIdentifier](#) [method](#)), [14](#)  
[load\\_model\(\)](#) ([framenet\\_tools.frame\\_identification.frameidnetwork.FrameIDNetwork](#) [method](#)), [15](#)  
[load\\_model\(\)](#) ([framenet\\_tools.span\\_identification.spanidentifier.SpanIdentifier](#) [method](#)), [17](#)  
[load\\_model\(\)](#) ([framenet\\_tools.span\\_identification.spanidnetwork.SpanIDNetwork](#) [method](#)), [20](#)  
[load\\_pkl\\_from\\_path\(\)](#) (in [module](#) [framenet\\_tools.utils.static\\_utils](#)), [24](#)  
[loaded\(\)](#) ([framenet\\_tools.data\\_handler.reader.DataReader](#) [method](#)), [10](#)  
**M**  
[main\(\)](#) (in [module](#) [framenet\\_tools.main](#)), [26](#)  
**P**  
[paths\\_to\\_string\(\)](#) ([framenet\\_tools.config.ConfigManager](#) [method](#)), [25](#)  
[Pipeline](#) (class in [framenet\\_tools.pipeline](#)), [27](#)  
[PipelineStage](#) (class in [framenet\\_tools.pipelinestage](#)), [28](#)  
[pos\\_to\\_int\(\)](#) (in [module](#) [framenet\\_tools.utils.static\\_utils](#)), [24](#)  
[PosTagger](#) (class in [framenet\\_tools.utils.postagger](#)), [22](#)  
[predict\(\)](#) ([framenet\\_tools.frame\\_identification.frameidnetwork.FrameIDNetwork](#) [method](#)), [16](#)  
[predict\(\)](#) ([framenet\\_tools.pipeline.Pipeline](#) [method](#)), [27](#)  
[predict\(\)](#) ([framenet\\_tools.pipelinestage.PipelineStage](#) [method](#)), [28](#)  
[predict\(\)](#) ([framenet\\_tools.span\\_identification.spanidnetwork.SpanIDNetwork](#) [method](#)), [20](#)  
[predict\(\)](#) ([framenet\\_tools.stages.feeID.FeeID](#) [method](#)), [20](#)  
[predict\(\)](#) ([framenet\\_tools.stages.frameID.FrameID](#) [method](#)), [21](#)  
[predict\(\)](#) ([framenet\\_tools.stages.roleID.RoleID](#) [method](#)), [21](#)  
[predict\(\)](#) ([framenet\\_tools.stages.spanID.SpanID](#) [method](#)), [22](#)  
[predict\\_fees\(\)](#) ([framenet\\_tools.fee\\_identification.feeidentifier.FeeIdentifier](#) [method](#)), [12](#)  
[predict\\_fees\\_old\(\)](#) ([framenet\\_tools.fee\\_identification.feeidentifier.FeeIdentifier](#) [method](#)), [13](#)  
[predict\\_roles\(\)](#) ([framenet\\_tools.role\\_identification.roleidentifier.RoleIdentifier](#) [method](#)), [16](#)  
[predict\\_spans\(\)](#) ([framenet\\_tools.span\\_identification.spanidentifier.SpanIdentifier](#) [method](#)), [17](#)  
[prepare\\_dataset\(\)](#) ([framenet\\_tools.frame\\_identification.frameidentifier.FrameIdentifier](#) [method](#)), [14](#)  
[prepare\\_dataset\(\)](#) ([framenet\\_tools.span\\_identification.spanidentifier.SpanIdentifier](#) [method](#)), [18](#)  
[print\\_dict\\_to\\_txt\(\)](#) (in [module](#) [framenet\\_tools.utils.static\\_utils](#)), [24](#)  
**Q**  
[query\(\)](#) ([framenet\\_tools.fee\\_identification.feeidentifier.FeeIdentifier](#) [method](#)), [13](#)  
[query\(\)](#) ([framenet\\_tools.frame\\_identification.frameidentifier.FrameIdentifier](#) [method](#)), [14](#)  
[query\(\)](#) ([framenet\\_tools.frame\\_identification.frameidnetwork.FrameIDNetwork](#) [method](#)), [16](#)  
[query\(\)](#) ([framenet\\_tools.span\\_identification.spanidentifier.SpanIdentifier](#) [method](#)), [18](#)  
[query\\_all\(\)](#) ([framenet\\_tools.span\\_identification.spanidentifier.SpanIdentifier](#) [method](#)), [18](#)  
[query\\_confidence\(\)](#) ([framenet\\_tools.frame\\_identification.frameidentifier.FrameIdentifier](#) [method](#)), [14](#)  
[query\\_nn\(\)](#) ([framenet\\_tools.span\\_identification.spanidentifier.SpanIdentifier](#) [method](#)), [18](#)  
[query\\_static\(\)](#) ([framenet\\_tools.span\\_identification.spanidentifier.SpanIdentifier](#) [method](#)), [18](#)  
**R**  
[RawReader](#) (class in [framenet\\_tools.data\\_handler.rawreader](#)), [16](#)

9

**T**

`read_data()` (`framenet_tools.data_handler.semaforreader.SemaforReader` (class in `framenet_tools.data_handler.semaforreader`), 11)

`read_data()` (`framenet_tools.data_handler.semevalreader.SemevalReader` (class in `framenet_tools.data_handler.semevalreader`), 11)

`read_frame_embeddings()` (`framenet_tools.data_handler.frame_embedding_manager.FrameEmbeddingManager` (class in `framenet_tools.data_handler.frame_embedding_manager`), 8)

`read_raw_text()` (`framenet_tools.data_handler.rawreader.RawReader` (class in `framenet_tools.data_handler.rawreader`), 9)

`read_word_embeddings()` (`framenet_tools.data_handler.word_embedding_manager.WordEmbeddingManager` (class in `framenet_tools.data_handler.word_embedding_manager`), 12)

`reset_hidden()` (`framenet_tools.span_identification.spanidnetwork.SpanIdNetwork` (class in `framenet_tools.span_identification.spanidnetwork`), 20)

`RoleID` (class in `framenet_tools.stages.roleID`), 21

`RoleIdentifier` (class in `framenet_tools.role_identification.roleidentifier`), 16

**S**

`save_model()` (`framenet_tools.frame_identification.frameidentifier.FrameIdentifier` (class in `framenet_tools.frame_identification.frameidentifier`), 14)

`save_model()` (`framenet_tools.frame_identification.frameidnetwork.FrameIdNetwork` (class in `framenet_tools.frame_identification.frameidnetwork`), 16)

`save_model()` (`framenet_tools.span_identification.spanidentifier.SpanIdentifier` (class in `framenet_tools.span_identification.spanidentifier`), 19)

`save_model()` (`framenet_tools.span_identification.spanidnetwork.SpanIdNetwork` (class in `framenet_tools.span_identification.spanidnetwork`), 20)

`SemaforReader` (class in `framenet_tools.data_handler.semaforreader`), 10

`SemevalReader` (class in `framenet_tools.data_handler.semevalreader`), 11

`should_include_token()` (in module `framenet_tools.fee_identification.feeidentifier`), 13

`shuffle_concurrent_lists()` (in module `framenet_tools.utils.static_utils`), 24

`SpanID` (class in `framenet_tools.stages.spanID`), 22

`SpanIdentifier` (class in `framenet_tools.span_identification.spanidentifier`), 17

`SpanIdNetwork` (class in `framenet_tools.span_identification.spanidnetwork`), 19

`string_to_array()` (`framenet_tools.data_handler.frame_embedding_manager.FrameEmbeddingManager` (class in `framenet_tools.data_handler.frame_embedding_manager`), 8)

`string_to_array()` (`framenet_tools.data_handler.word_embedding_manager.WordEmbeddingManager` (class in `framenet_tools.data_handler.word_embedding_manager`), 12)

**W**

`write_predictions()` (`framenet_tools.frame_identification.frameidentifier.FrameIdentifier` (class in `framenet_tools.frame_identification.frameidentifier`), 15)