

---

# **TUPA Documentation**

***Release 1.4.2***

**Daniel Hershcovich**

**Jan 14, 2020**



---

## Contents:

---

<b>1</b>	<b>Getting Started</b>	<b>3</b>
<b>2</b>	<b>tupa.parse Module</b>	<b>5</b>
2.1	Functions . . . . .	5
2.2	Classes . . . . .	7
2.3	Class Inheritance Diagram . . . . .	12
<b>3</b>	<b>tupa.action Module</b>	<b>13</b>
3.1	Classes . . . . .	13
3.2	Class Inheritance Diagram . . . . .	15
<b>4</b>	<b>tupa.config Module</b>	<b>17</b>
4.1	Functions . . . . .	17
4.2	Classes . . . . .	17
4.3	Class Inheritance Diagram . . . . .	18
<b>5</b>	<b>tupa.labels Module</b>	<b>19</b>
5.1	Classes . . . . .	19
5.2	Class Inheritance Diagram . . . . .	20
<b>6</b>	<b>tupa.model Module</b>	<b>21</b>
6.1	Functions . . . . .	21
6.2	Classes . . . . .	21
6.3	Class Inheritance Diagram . . . . .	26
<b>7</b>	<b>tupa.model_util Module</b>	<b>27</b>
7.1	Functions . . . . .	27
7.2	Classes . . . . .	28
7.3	Class Inheritance Diagram . . . . .	31
<b>8</b>	<b>tupa.oracle Module</b>	<b>33</b>
8.1	Functions . . . . .	33
8.2	Classes . . . . .	33
8.3	Class Inheritance Diagram . . . . .	35
<b>9</b>	<b>Indices and tables</b>	<b>37</b>
	<b>Python Module Index</b>	<b>39</b>



For more information about how to use this library, see the *API Documentation*.



# CHAPTER 1

---

## Getting Started

---

To parse text to UCCA passages, download a model file from [the latest release](#), extract it, and use the following code template:

```
from tupa.parse import Parser
from ucca.convert import from_text
parser = Parser("models/ucca_bilstm")
for passage in parser.parse(from_text(...)):
```

Each passage instantiates the `ucca.core.Passage` class.





### 2.1 Functions

<i>average_f1</i> (scores[, eval_type])	
<i>filter_passages_for_bert</i> (passages, args)	
<i>from_text_format</i> (*args, **kwargs)	
<i>generate_and_len</i> (it)	
<i>get_eval_type</i> (scores)	
<i>get_output_converter</i> (out_format[, default])	
<i>glob</i> (pathname, *[, recursive])	Return a list of paths matching a pathname pattern.
<i>main</i> ()	
<i>main_generator</i> ()	
<i>percents_str</i> (part, total[, infix, fraction])	
<i>print_scores</i> (scores, filename[, prefix, ...])	
<i>read_passages</i> (args, files)	
<i>set_traceback_listener</i> ([sig])	
<i>single_to_iter</i> (it)	
<i>to_lower_case</i> (passages)	
<i>train_test</i> (train_passages, dev_passages, ...)	Train and test parser on given passage :param train_passages: passage to train on :param dev_passages: passages to evaluate on every iteration :param test_passages: passages to test on after training :param args: extra argument :param model_suffix: string to append to model filename before file extension :return: generator of Scores objects: dev scores for each training iteration (if given dev), and finally test scores

#### 2.1.1 average\_f1

tupa.parse.**average\_f1** (scores, eval\_type=None)

### 2.1.2 filter\_passages\_for\_bert

`tupa.parse.filter_passages_for_bert` (*passages, args*)

### 2.1.3 from\_text\_format

`tupa.parse.from_text_format` (*\*args, \*\*kwargs*)

### 2.1.4 generate\_and\_len

`tupa.parse.generate_and_len` (*it*)

### 2.1.5 get\_eval\_type

`tupa.parse.get_eval_type` (*scores*)

### 2.1.6 get\_output\_converter

`tupa.parse.get_output_converter` (*out\_format, default=None*)

### 2.1.7 main

`tupa.parse.main` ()

### 2.1.8 main\_generator

`tupa.parse.main_generator` ()

### 2.1.9 percents\_str

`tupa.parse.percents_str` (*part, total, infix=", fraction=True*)

### 2.1.10 print\_scores

`tupa.parse.print_scores` (*scores, filename, prefix=None, prefix\_title=None*)

### 2.1.11 read\_passages

`tupa.parse.read_passages` (*args, files*)

### 2.1.12 single\_to\_iter

`tupa.parse.single_to_iter` (*it*)

### 2.1.13 to\_lower\_case

`tupa.parse.to_lower_case` (*passages*)

### 2.1.14 train\_test

`tupa.parse.train_test` (*train\_passages, dev\_passages, test\_passages, args, model\_suffix=""*)

Train and test parser on given passage :param train\_passages: passage to train on :param dev\_passages: passages to evaluate on every iteration :param test\_passages: passages to test on after training :param args: extra argument :param model\_suffix: string to append to model filename before file extension :return: generator of Scores objects: dev scores for each training iteration (if given dev), and finally test scores

## 2.2 Classes

<code>AbstractParser</code> ( <i>config, models[, training, ...]</i> )	
<code>BatchParser</code> (*args, **kwargs)	Parser for a single training iteration or single pass over dev/test passages
<code>ClassifierProperty</code>	An enumeration.
<code>Enum</code>	Generic enumeration.
<code>Iterations</code> (args)	
<code>Model</code> (filename[, config])	
<code>Oracle</code> ( <i>passage</i> )	Oracle to produce gold transition parses given UCCA passages To be used for creating training data for a transition-based UCCA parser :param passage gold passage to get the correct edges from
<code>ParseMode</code>	An enumeration.
<code>Parser</code> ([ <i>model_files, config, beam</i> ])	Main class to implement transition-based UCCA parser
<code>ParserException</code>	
<code>PassageParser</code> ( <i>passage, *args, **kwargs</i> )	Parser for a single passage, has a state and optionally an oracle
<code>State</code> ( <i>passage</i> )	The parser's state, responsible for applying actions and creating the final Passage :param passage: a Passage object to get the tokens from, and everything else if training
<code>defaultdict</code>	<code>defaultdict(default_factory[, ...])</code> -> dict with default factory
<code>partial</code>	<code>partial(func, *args, **keywords)</code> - new function with partial application of the given arguments and keywords.

### 2.2.1 AbstractParser

**class** `tupa.parse.AbstractParser` (*config, models, training=False, evaluation=False*)  
 Bases: `object`

#### Attributes Summary

---

*duration*

---

*model*

---

### Methods Summary

---

*tokens\_per\_second()*

---

### Attributes Documentation

**duration****model**

### Methods Documentation

**tokens\_per\_second()**

## 2.2.2 BatchParser

**class** `tupa.parse.BatchParser(*args, **kwargs)`Bases: `tupa.parse.AbstractParser`

Parser for a single training iteration or single pass over dev/test passages

### Methods Summary

---

*add\_progress\_bar*(it[, total, display])

---

*parse*(passages[, display, write, accuracies])

---

*summary*()

---

*time\_per\_passage*()

---

*update\_counts*(parser)

---

### Methods Documentation

**add\_progress\_bar** (*it*, *total=None*, *display=True*)**parse** (*passages*, *display=True*, *write=False*, *accuracies=None*)**summary** ()**time\_per\_passage** ()**update\_counts** (*parser*)

## 2.2.3 ParseMode

**class** `tupa.parse.ParseMode`Bases: `enum.Enum`

An enumeration.

## Attributes Summary

<code>dev</code>
<code>test</code>
<code>train</code>

## Attributes Documentation

`dev = 2`

`test = 3`

`train = 1`

## 2.2.4 Parser

**class** `tupa.parse.Parser` (*model\_files=()*, *config=None*, *beam=1*)

Bases: `tupa.parse.AbstractParser`

Main class to implement transition-based UCCA parser

## Methods Summary

<code>eval</code> ( <i>passages</i> , <i>mode</i> , <i>scores_filename</i> [, <i>display</i> ])	
<code>eval_and_save</code> ([ <i>last</i> , <i>finished_epoch</i> ])	
<code>init_train</code> ()	
<code>parse</code> ( <i>passages</i> [, <i>mode</i> , <i>evaluate</i> , <i>display</i> , <i>write</i> ])	Parse given passages :param passages: iterable of passages to parse :param mode: ParseMode value.
<code>print_config</code> ()	
<code>save</code> ( <i>model</i> )	
<code>train</code> ([ <i>passages</i> , <i>dev</i> , <i>test</i> , <i>iterations</i> ])	Train parser on given passages :param passages: iterable of passages to train on :param dev: iterable of passages to tune on :param test: iterable of passages that would be tested on after train finished :param iterations: iterable of Iterations objects whose i attributes are the number of iterations to perform

## Methods Documentation

**eval** (*passages*, *mode*, *scores\_filename*, *display=True*)

**eval\_and\_save** (*last=False*, *finished\_epoch=False*)

**init\_train** ()

**parse** (*passages*, *mode=<ParseMode.test: 3>*, *evaluate=False*, *display=True*, *write=False*)

Parse given passages :param passages: iterable of passages to parse :param mode: ParseMode value.

If train, use oracle to train on given passages. Otherwise, just parse with classifier.

### Parameters

- **evaluate** – whether to evaluate parsed passages with respect to given ones. Only possible when given passages are annotated.

- **display** – whether to display information on each parsed passage
- **write** – whether to write output passages to file

**Returns** generator of parsed passages (or in train mode, the original ones), or, if evaluation=True, of pairs of (Passage, Scores).

**print\_config()**

**save** (*model*)

**train** (*passages=None, dev=None, test=None, iterations=1*)

Train parser on given passages :param passages: iterable of passages to train on :param dev: iterable of passages to tune on :param test: iterable of passages that would be tested on after train finished :param iterations: iterable of Iterations objects whose i attributes are the number of iterations to perform

## 2.2.5 ParserException

**exception** `tupa.parse.ParserException`

## 2.2.6 PassageParser

**class** `tupa.parse.PassageParser` (*passage, \*args, \*\*kwargs*)

Bases: `tupa.parse.AbstractParser`

Parser for a single passage, has a state and optionally an oracle

### Attributes Summary

---

`accuracy_str`

---

`num_tokens`

---

### Methods Summary

---

`check_loop()`

Check if the current state has already occurred, indicating a loop

---

`choose(true[, axis, name])`

---

`correct(axis, label, pred, scores, true, ...)`

---

`evaluate([mode])`

---

`finish(status[, display, write, accuracies])`

---

`generate_descending(scores)`

---

`get_true_actions()`

---

`get_true_label(node)`

---

`init()`

---

`label_node([action])`

---

`parse([display, write, accuracies])`

---

`parse_internal()`

Internal method to parse a single passage.

Continued on next page

Table 9 – continued from previous page

<code>predict(scores, values[, is_valid])</code>	Choose action/label based on classifier Usually the best action/label is valid, so max is enough to choose it in $O(n)$ time Otherwise, sorts all the other scores to choose the best valid one in $O(n \lg n)$ :return: valid action/label with maximum probability according to classifier
<code>verify(guessed, ref)</code>	Compare predicted passage to true passage and raise an exception if they differ :param ref: true passage :param guessed: predicted passage to compare

## Attributes Documentation

`accuracy_str`

`num_tokens`

## Methods Documentation

`check_loop()`

Check if the current state has already occurred, indicating a loop

`choose(true, axis=None, name='action')`

`correct(axis, label, pred, scores, true, true_keys)`

`evaluate(mode=<ParseMode.test: 3>)`

`finish(status, display=True, write=False, accuracies=None)`

`static generate_descending(scores)`

`get_true_actions()`

`get_true_label(node)`

`init()`

`label_node(action=None)`

`parse(display=True, write=False, accuracies=None)`

`parse_internal()`

Internal method to parse a single passage. If training, use oracle to train on given passages. Otherwise just parse with classifier.

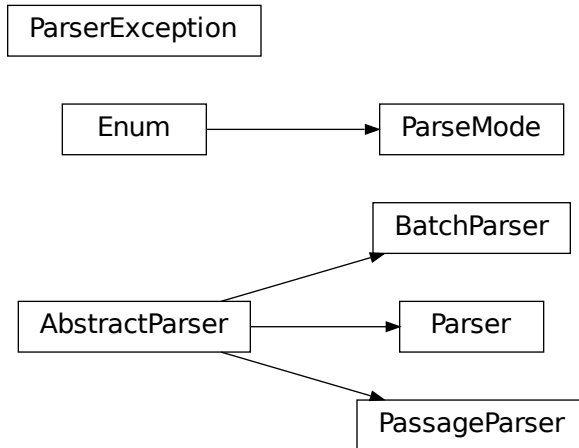
`static predict(scores, values, is_valid=None)`

Choose action/label based on classifier Usually the best action/label is valid, so max is enough to choose it in  $O(n)$  time Otherwise, sorts all the other scores to choose the best valid one in  $O(n \lg n)$  :return: valid action/label with maximum probability according to classifier

`verify(guessed, ref)`

Compare predicted passage to true passage and raise an exception if they differ :param ref: true passage :param guessed: predicted passage to compare

## 2.3 Class Inheritance Diagram





## 3.1 Classes

---

*Action*(action\_type[, tag, orig\_edge, ...])  
*Actions*([actions, size])  
 Labels(size)

---

### 3.1.1 Action

**class** tupa.action.**Action**(action\_type, tag=None, orig\_edge=None, orig\_node=None, oracle=None, id\_=None)  
 Bases: dict

#### Attributes Summary

---

*is\_swap*  
*remote*  
*type\_to\_id*

---

#### Methods Summary

---

*\_\_call\_\_*(\*args, \*\*kwargs) Call self as a function.  
*apply*()  
*is\_type*(\*others)

---

#### Attributes Documentation

**is\_swap**

---

`remote`

`type_to_id = {'FINISH': 11, 'IMPLICIT': 3, 'LABEL': 4, 'LEFT-EDGE': 6, 'LEFT-REMOTE': 11, 'RIGHT-EDGE': 6, 'RIGHT-REMOTE': 11}`

### Methods Documentation

`__call__(*args, **kwargs)`  
Call self as a function.

`apply()`

`is_type(*others)`

## 3.1.2 Actions

**class** `tupa.action.Actions` (*actions=None, size=None*)

Bases: `tupa.labels.Labels`

### Attributes Summary

<i>Finish</i>
<i>Implicit</i>
<i>Label</i>
<i>LeftEdge</i>
<i>LeftRemote</i>
<i>Node</i>
<i>Reduce</i>
<i>RemoteNode</i>
<i>RightEdge</i>
<i>RightRemote</i>
<i>Shift</i>
<i>Swap</i>
<i>all</i>
<i>ids</i>

### Methods Summary

<i>generate_id</i> (action[, create])
<i>init</i> ()

### Attributes Documentation

`Finish = {'action_type': 'FINISH', 'tag': None}`

`Implicit = {'action_type': 'IMPLICIT', 'tag': None}`

`Label = {'action_type': 'LABEL', 'tag': None}`

`LeftEdge = {'action_type': 'LEFT-EDGE', 'tag': None}`

`LeftRemote = {'action_type': 'LEFT-REMOTE', 'tag': None}`

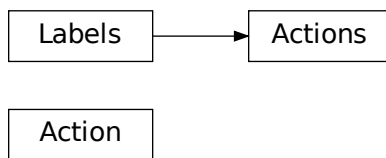
`Node = {'action_type': 'NODE', 'tag': None}`

```
Reduce = {'action_type': 'REDUCE', 'tag': None}
RemoteNode = {'action_type': 'REMOTE-NODE', 'tag': None}
RightEdge = {'action_type': 'RIGHT-EDGE', 'tag': None}
RightRemote = {'action_type': 'RIGHT-REMOTE', 'tag': None}
Shift = {'action_type': 'SHIFT', 'tag': None}
Swap = {'action_type': 'SWAP', 'tag': None}
all
ids
```

### Methods Documentation

```
generate_id(action, create=True)
init()
```

## 3.2 Class Inheritance Diagram





### 4.1 Functions

---

<code>add_param_arguments([ap, arg_default])</code>	
<code>deepcopy(x[, memo, _nil])</code>	Deep copy operation on arbitrary Python objects.
<code>load_enum(filename)</code>	

---

#### 4.1.1 add\_param\_arguments

`tupa.config.add_param_arguments` (*ap=None, arg\_default=None*)

### 4.2 Classes

---

<code>CategoricalParameter(values, string)</code>
<code>Hyperparams(parent[, shared])</code>
<code>HyperparamsInitializer([name])</code>
<code>Iterations(args)</code>

---

#### 4.2.1 Hyperparams

**class** `tupa.config.Hyperparams` (*parent, shared=None, \*\*kwargs*)  
 Bases: `object`

##### Methods Summary

---

`items()`

---

### Methods Documentation

`items()`

## 4.2.2 HyperparamsInitializer

```
class tupa.config.HyperparamsInitializer(name=None, *args, **kwargs)
    Bases: object
```

### Methods Summary

---

`action(args)`

---

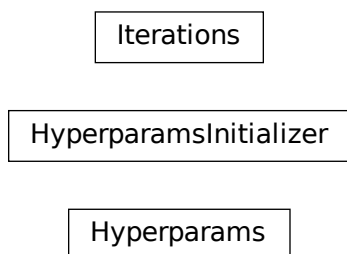
### Methods Documentation

`classmethod action(args)`

## 4.2.3 Iterations

```
class tupa.config.Iterations(args)
    Bases: object
```

## 4.3 Class Inheritance Diagram



## 5.1 Classes

---

*Labels*(size)

---

### 5.1.1 Labels

**class** tupa.labels.**Labels** (*size*)  
Bases: `object`

#### Attributes Summary

---

*all*

---

#### Methods Summary

---

*load*(all\_size)

---

*save*([skip])

---

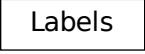
#### Attributes Documentation

**all**

#### Methods Documentation

**load** (*all\_size*)  
**save** (*skip=False*)

## 5.2 Class Inheritance Diagram



Labels



### 6.1 Functions

<code>load_json(filename)</code>	Load dictionary from JSON file :param filename: file to read from
<code>remove_backup(*filenames)</code>	
<code>save_json(filename, d)</code>	Save dictionary to JSON file :param filename: file to write to :param d: dictionary to save

#### 6.1.1 load\_json

`tupa.model.load_json(filename)`  
 Load dictionary from JSON file :param filename: file to read from

#### 6.1.2 remove\_backup

`tupa.model.remove_backup(*filenames)`

#### 6.1.3 save\_json

`tupa.model.save_json(filename, d)`  
 Save dictionary to JSON file :param filename: file to write to :param d: dictionary to save

### 6.2 Classes

<code>Actions([actions, size])</code>	
---------------------------------------	--

---

Continued on next page

Table 2 – continued from previous page

<i>AutoIncrementDict</i> ([size, keys, d, unknown])	DefaultOrderedDict that returns an auto-incrementing index for new keys
<i>Classifier</i> (config, labels[, input_params])	Interface for classifier used by the parser.
<i>ClassifierProperty</i>	An enumeration.
Enum	Generic enumeration.
FeatureParameters(suffix, dim, size[, ...])	
<i>Model</i> (filename[, config])	
OrderedDict	Dictionary that remembers insertion order
<i>ParameterDefinition</i> (args, name, attr_to_arg)	
<i>UnknownDict</i> ([d])	DefaultOrderedDict that has a single default value for missing keys

## 6.2.1 AutoIncrementDict

**class** tupa.model.AutoIncrementDict (*size=None, keys=(), d=None, unknown=0*)

Bases: *tupa.model\_util.DefaultOrderedDict*

DefaultOrderedDict that returns an auto-incrementing index for new keys

### Methods Summary

---

*first\_items*([n])

---

### Methods Documentation

**first\_items** (*n=3*)

## 6.2.2 ClassifierProperty

**class** tupa.model.ClassifierProperty

Bases: *enum.Enum*

An enumeration.

### Attributes Summary

---

*require\_init\_features*

---

*trainable\_after\_saving*

---

*update\_only\_on\_error*

---

### Attributes Documentation

**require\_init\_features** = 2

**trainable\_after\_saving** = 3

**update\_only\_on\_error** = 1

### 6.2.3 Model

**class** `tupa.model.Model` (*filename*, *config=None*, \*args, \*\*kwargs)  
 Bases: `object`

#### Attributes Summary

---

*actions*

---



---

*classifier\_properties*

---



---

*formats*

---



---

*is\_neural\_network*

---



---

*is\_retrainable*

---

#### Methods Summary

---

*all\_params()*

---



---

*finalize*(finished\_epoch)

---

 Copy model, finalizing features (new values will not be added during subsequent use) and classifier (update it) :param finished\_epoch: whether this is the end of an epoch (or just intermediate checkpoint), for bookkeeping :return: a copy of this model with a new feature extractor and classifier (actually classifier may be the same)

---



---

*init\_actions()*

---



---

*init\_features*(state, train)

---



---

*init\_model*([axis, lang, init\_params])

---



---

*init\_node\_labels()*

---



---

*init\_param*(key)

---



---

*load*([is\_finalized])

---

 Load the feature and classifier parameters from files :param is\_finalized: whether loaded model should be finalized, or allow feature values to be added subsequently

---



---

*load\_labels()*

---

 Copy classifier's labels to create new Actions/Labels objects Restoring from a model that was just loaded from file, or called by restore()

---



---

*node\_label\_param\_def*([args])

---



---

*param\_defs*([args, only\_node\_labels])

---



---

*restore*(model[, feature\_extractor, ...])

---

 Set all attributes to a reference to existing model, except labels, which will be copied.

---



---

*save*([save\_init])

---

 Save feature and classifier parameters to files

---



---

*score*(state, axis)

---



---

*set\_axis*(axis, lang)

---

#### Attributes Documentation

**actions**  
**classifier\_properties**  
**formats**  
**is\_neural\_network**

**is\_retrainable**

## Methods Documentation

**all\_params**()

**finalize**(*finished\_epoch*)

Copy model, finalizing features (new values will not be added during subsequent use) and classifier (update it) :param finished\_epoch: whether this is the end of an epoch (or just intermediate checkpoint), for bookkeeping :return: a copy of this model with a new feature extractor and classifier (actually classifier may be the same)

**init\_actions**()

**init\_features**(*state*, *train*)

**init\_model**(*axis=None*, *lang=None*, *init\_params=True*)

**init\_node\_labels**()

**init\_param**(*key*)

**load**(*is\_finalized=True*)

Load the feature and classifier parameters from files :param is\_finalized: whether loaded model should be finalized, or allow feature values to be added subsequently

**load\_labels**()

Copy classifier's labels to create new Actions/Labels objects Restoring from a model that was just loaded from file, or called by restore()

**node\_label\_param\_def**(*args=None*)

**param\_defs**(*args=None*, *only\_node\_labels=False*)

**restore**(*model*, *feature\_extractor=None*, *classifier=None*, *is\_finalized=None*)

Set all attributes to a reference to existing model, except labels, which will be copied. :param model: Model to restore :param feature\_extractor: optional FeatureExtractor to restore instead of model's :param classifier: optional Classifier to restore instead of model's :param is\_finalized: whether the restored model is finalized

**save**(*save\_init=False*)

Save feature and classifier parameters to files

**score**(*state*, *axis*)

**set\_axis**(*axis*, *lang*)

## 6.2.4 ParameterDefinition

**class** `tupa.model.ParameterDefinition`(*args*, *name*, *attr\_to\_arg*, *attr\_to\_val=None*)

Bases: `object`

### Attributes Summary

---

*dim\_arg*

---

*enabled*

---

*lang\_specific*

---

Continued on next page

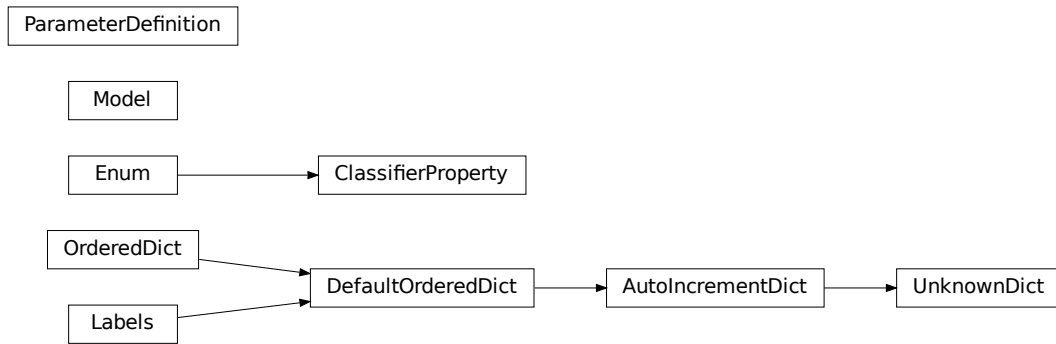
Table 7 – continued from previous page

<i>size_arg</i>
<b>Methods Summary</b>
<i>all_langs</i> (params)
<i>create_from_config</i> ([lang])
<i>get_args</i> (lang)
<i>key</i> ([lang])
<i>load_to_config</i> (params)

**Attributes Documentation****dim\_arg****enabled****lang\_specific****size\_arg****Methods Documentation****all\_langs** (params)**create\_from\_config** (lang=None)**get\_args** (lang)**key** (lang=None)**load\_to\_config** (params)**6.2.5 UnknownDict****class** tupa.model.UnknownDict (d=None)Bases: *tupa.model\_util.AutoIncrementDict*

DefaultOrderedDict that has a single default value for missing keys

## 6.3 Class Inheritance Diagram



## 7.1 Functions

<code>glob(pathname, *[, recursive])</code>	Return a list of paths matching a pathname pattern.
<code>jsonify(o)</code>	
<code>load_dict(filename)</code>	Load dictionary from Pickle file :param filename: file to read from
<code>load_enum(filename)</code>	
<code>load_json(filename)</code>	Load dictionary from JSON file :param filename: file to read from
<code>remove_backup(*filenames)</code>	
<code>remove_existing(*filenames)</code>	
<code>save_dict(filename, d)</code>	Save dictionary to Pickle file :param filename: file to write to :param d: dictionary to save
<code>save_json(filename, d)</code>	Save dictionary to JSON file :param filename: file to write to :param d: dictionary to save

### 7.1.1 jsonify

`tupa.model_util.jsonify(o)`

### 7.1.2 load\_dict

`tupa.model_util.load_dict(filename)`  
Load dictionary from Pickle file :param filename: file to read from

### 7.1.3 load\_enum

`tupa.model_util.load_enum(filename)`

### 7.1.4 load\_json

`tupa.model_util.load_json(filename)`  
Load dictionary from JSON file :param filename: file to read from

### 7.1.5 remove\_backup

`tupa.model_util.remove_backup(*filenames)`

### 7.1.6 remove\_existing

`tupa.model_util.remove_existing(*filenames)`

### 7.1.7 save\_dict

`tupa.model_util.save_dict(filename, d)`  
Save dictionary to Pickle file :param filename: file to write to :param d: dictionary to save

### 7.1.8 save\_json

`tupa.model_util.save_json(filename, d)`  
Save dictionary to JSON file :param filename: file to write to :param d: dictionary to save

## 7.2 Classes

<i>AutoIncrementDict</i> ([size, keys, d, unknown])	DefaultOrderedDict that returns an auto-incrementing index for new keys
<i>Counter</i> (**kwds)	Dict subclass for counting hashable items.
<i>DefaultOrderedDict</i> ([default_factory, size])	
<i>DropoutDict</i> ([d, dropout, size, keys, min_count])	UnknownDict that sometimes returns the unknown value even for existing keys
<i>IdentityVocab</i> ()	
<i>KeyBasedDefaultDict</i>	
<i>Labels</i> (size)	
<i>Lexeme</i> (index, text)	
<i>OrderedDict</i>	Dictionary that remembers insertion order
<i>Strings</i> (vocab)	
<i>UnknownDict</i> ([d])	DefaultOrderedDict that has a single default value for missing keys
<i>Vocab</i> (tuples)	
<i>defaultdict</i>	<code>defaultdict(default_factory[, ...])</code> -> dict with default factory
<i>itemgetter</i>	<code>itemgetter(item, ...)</code> -> itemgetter object



### 7.2.1 AutoIncrementDict

**class** `tupa.model_util.AutoIncrementDict` (*size=None, keys=(), d=None, unknown=0*)

Bases: `tupa.model_util.DefaultOrderedDict`

DefaultOrderedDict that returns an auto-incrementing index for new keys

#### Methods Summary

---

*first\_items*([n])

---

#### Methods Documentation

**first\_items** (*n=3*)

### 7.2.2 DefaultOrderedDict

**class** `tupa.model_util.DefaultOrderedDict` (*default\_factory=None, \*args, size=None, \*\*kwargs*)

Bases: `collections.OrderedDict, tupa.labels.Labels`

#### Attributes Summary

---

*all*

---

#### Methods Summary

---

*copy*()

---

#### Attributes Documentation

**all**

#### Methods Documentation

**copy** () → a shallow copy of od

### 7.2.3 DropoutDict

**class** `tupa.model_util.DropoutDict` (*d=None, dropout=0, size=None, keys=(), min\_count=1*)

Bases: `tupa.model_util.AutoIncrementDict`

UnknownDict that sometimes returns the unknown value even for existing keys

### 7.2.4 IdentityVocab

**class** `tupa.model_util.IdentityVocab`  
Bases: `tupa.model_util.Vocab`

### 7.2.5 KeyBasedDefaultDict

**class** `tupa.model_util.KeyBasedDefaultDict`  
Bases: `collections.defaultdict`

### 7.2.6 Lexeme

**class** `tupa.model_util.Lexeme` (*index, text*)  
Bases: `object`

### 7.2.7 Strings

**class** `tupa.model_util.Strings` (*vocab*)  
Bases: `object`

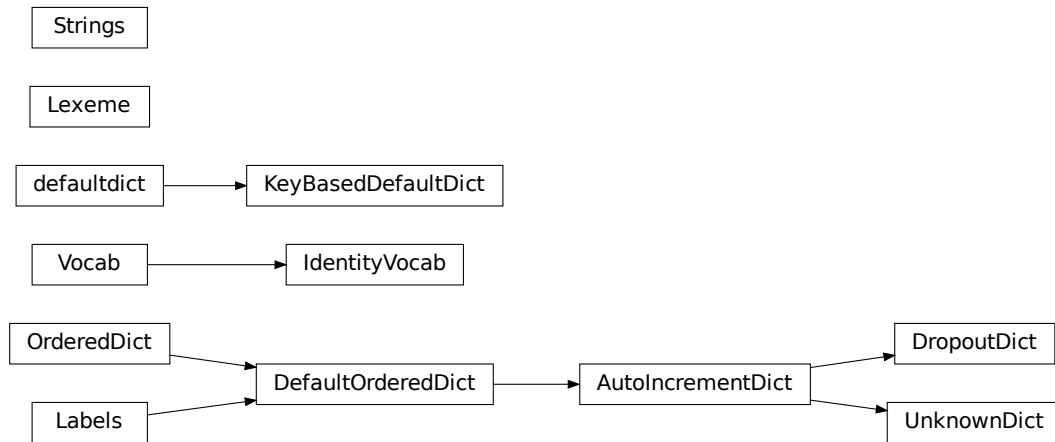
### 7.2.8 UnknownDict

**class** `tupa.model_util.UnknownDict` (*d=None*)  
Bases: `tupa.model_util.AutoIncrementDict`  
DefaultOrderedDict that has a single default value for missing keys

### 7.2.9 Vocab

**class** `tupa.model_util.Vocab` (*tuples*)  
Bases: `dict`

## 7.3 Class Inheritance Diagram





## 8.1 Functions

---

*is\_implicit\_node*(node)

---

*is\_remote\_edge*(edge)

---

*is\_terminal\_edge*(edge)

---

### 8.1.1 is\_implicit\_node

tupa.oracle.**is\_implicit\_node** (*node*)

### 8.1.2 is\_remote\_edge

tupa.oracle.**is\_remote\_edge** (*edge*)

### 8.1.3 is\_terminal\_edge

tupa.oracle.**is\_terminal\_edge** (*edge*)

## 8.2 Classes

---

Actions([actions, size])

---

InvalidActionError(\*args[, is\_type])

---

Continued on next page

Table 2 – continued from previous page

<code>Oracle</code> ( <i>passage</i> )	Oracle to produce gold transition parses given UCCA passages To be used for creating training data for a transition-based UCCA parser :param <i>passage</i> gold passage to get the correct edges from
--	--

### 8.2.1 Oracle

**class** `tupa.oracle.Oracle` (*passage*)

Bases: `object`

Oracle to produce gold transition parses given UCCA passages To be used for creating training data for a transition-based UCCA parser :param *passage* gold passage to get the correct edges from

#### Methods Summary

<code>action</code> ( <i>edge</i> [, <i>kind</i> , <i>direction</i> ])	
<code>generate_actions</code> ( <i>state</i> )	Determine all zero-cost action according to current state :param <i>state</i> : current State of the parser :return: generator of Action items to perform
<code>generate_log</code> ( <i>invalid</i> , <i>state</i> )	
<code>get_actions</code> ( <i>state</i> , <i>all_actions</i> [, <i>create</i> ])	Determine all zero-cost action according to current state Asserts that the returned action is valid before returning :param <i>state</i> : current State of the parser :param <i>all_actions</i> : Actions object used to map actions to IDs :param <i>create</i> : whether to create new actions if they do not exist yet :return: dict of action ID to Action
<code>get_label</code> ( <i>state</i> , <i>node</i> )	
<code>need_label</code> ( <i>node</i> )	
<code>remove</code> ( <i>edge</i> [, <i>node</i> ])	
<code>str</code> ( <i>sep</i> )	

#### Methods Documentation

**action** (*edge*, *kind*=None, *direction*=None)

**generate\_actions** (*state*)

Determine all zero-cost action according to current state :param *state*: current State of the parser :return: generator of Action items to perform

**generate\_log** (*invalid*, *state*)

**get\_actions** (*state*, *all\_actions*, *create*=True)

Determine all zero-cost action according to current state Asserts that the returned action is valid before returning :param *state*: current State of the parser :param *all\_actions*: Actions object used to map actions to IDs :param *create*: whether to create new actions if they do not exist yet :return: dict of action ID to Action

**get\_label** (*state*, *node*)

**need\_label** (*node*)

**remove** (*edge*, *node*=None)

**str** (*sep*)

## 8.3 Class Inheritance Diagram



```
classDiagram
    class Oracle
```

A UML class diagram consisting of a single rectangular box with the text "Oracle" centered inside. The box has a thin black border.





## CHAPTER 9

---

### Indices and tables

---

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



### t

- `tupa.action`, 13
- `tupa.config`, 17
- `tupa.labels`, 19
- `tupa.model`, 21
- `tupa.model_util`, 27
- `tupa.oracle`, 33
- `tupa.parse`, 5



## Symbols

`__call__()` (*tupa.action.Action* method), 14

## A

*AbstractParser* (class in *tupa.parse*), 7  
*accuracy\_str* (*tupa.parse.PassageParser* attribute), 11

*Action* (class in *tupa.action*), 13

*action()* (*tupa.config.HyperparamsInitializer* class method), 18

*action()* (*tupa.oracle.Oracle* method), 34

*Actions* (class in *tupa.action*), 14

*actions* (*tupa.model.Model* attribute), 23

*add\_param\_arguments()* (in module *tupa.config*), 17

*add\_progress\_bar()* (*tupa.parse.BatchParser* method), 8

*all* (*tupa.action.Actions* attribute), 15

*all* (*tupa.labels.Labels* attribute), 19

*all* (*tupa.model\_util.DefaultOrderedDict* attribute), 29

*all\_langs()* (*tupa.model.ParameterDefinition* method), 25

*all\_params()* (*tupa.model.Model* method), 24

*apply()* (*tupa.action.Action* method), 14

*AutoIncrementDict* (class in *tupa.model*), 22

*AutoIncrementDict* (class in *tupa.model\_util*), 29

*average\_f1()* (in module *tupa.parse*), 5

## B

*BatchParser* (class in *tupa.parse*), 8

## C

*check\_loop()* (*tupa.parse.PassageParser* method), 11

*choose()* (*tupa.parse.PassageParser* method), 11

*classifier\_properties* (*tupa.model.Model* attribute), 23

*ClassifierProperty* (class in *tupa.model*), 22

*copy()* (*tupa.model\_util.DefaultOrderedDict* method), 29

*correct()* (*tupa.parse.PassageParser* method), 11

*create\_from\_config()* (*tupa.model.ParameterDefinition* method), 25

## D

*DefaultOrderedDict* (class in *tupa.model\_util*), 29

*dev* (*tupa.parse.ParseMode* attribute), 9

*dim\_arg* (*tupa.model.ParameterDefinition* attribute), 25

*DropoutDict* (class in *tupa.model\_util*), 29

*duration* (*tupa.parse.AbstractParser* attribute), 8

## E

*enabled* (*tupa.model.ParameterDefinition* attribute), 25

*eval()* (*tupa.parse.Parser* method), 9

*eval\_and\_save()* (*tupa.parse.Parser* method), 9

*evaluate()* (*tupa.parse.PassageParser* method), 11

## F

*filter\_passages\_for\_bert()* (in module *tupa.parse*), 6

*finalize()* (*tupa.model.Model* method), 24

*Finish* (*tupa.action.Actions* attribute), 14

*finish()* (*tupa.parse.PassageParser* method), 11

*first\_items()* (*tupa.model.AutoIncrementDict* method), 22

*first\_items()* (*tupa.model\_util.AutoIncrementDict* method), 29

*formats* (*tupa.model.Model* attribute), 23

*from\_text\_format()* (in module *tupa.parse*), 6

## G

*generate\_actions()* (*tupa.oracle.Oracle* method), 34

*generate\_and\_len()* (in module *tupa.parse*), 6

`generate_descending()`  
(*tupa.parse.PassageParser static method*), 11

`generate_id()` (*tupa.action.Actions method*), 15

`generate_log()` (*tupa.oracle.Oracle method*), 34

`get_actions()` (*tupa.oracle.Oracle method*), 34

`get_args()` (*tupa.model.ParameterDefinition method*), 25

`get_eval_type()` (*in module tupa.parse*), 6

`get_label()` (*tupa.oracle.Oracle method*), 34

`get_output_converter()` (*in module tupa.parse*), 6

`get_true_actions()` (*tupa.parse.PassageParser method*), 11

`get_true_label()` (*tupa.parse.PassageParser method*), 11

## H

`Hyperparams` (*class in tupa.config*), 17

`HyperparamsInitializer` (*class in tupa.config*), 18

## I

`IdentityVocab` (*class in tupa.model\_util*), 30

`ids` (*tupa.action.Actions attribute*), 15

`Implicit` (*tupa.action.Actions attribute*), 14

`init()` (*tupa.action.Actions method*), 15

`init()` (*tupa.parse.PassageParser method*), 11

`init_actions()` (*tupa.model.Model method*), 24

`init_features()` (*tupa.model.Model method*), 24

`init_model()` (*tupa.model.Model method*), 24

`init_node_labels()` (*tupa.model.Model method*), 24

`init_param()` (*tupa.model.Model method*), 24

`init_train()` (*tupa.parse.Parser method*), 9

`is_implicit_node()` (*in module tupa.oracle*), 33

`is_neural_network` (*tupa.model.Model attribute*), 23

`is_remote_edge()` (*in module tupa.oracle*), 33

`is_retrainable` (*tupa.model.Model attribute*), 23

`is_swap` (*tupa.action.Action attribute*), 13

`is_terminal_edge()` (*in module tupa.oracle*), 33

`is_type()` (*tupa.action.Action method*), 14

`items()` (*tupa.config.Hyperparams method*), 18

`Iterations` (*class in tupa.config*), 18

## J

`jsonify()` (*in module tupa.model\_util*), 27

## K

`key()` (*tupa.model.ParameterDefinition method*), 25

`KeyBasedDefaultDict` (*class in tupa.model\_util*), 30

## L

`Label` (*tupa.action.Actions attribute*), 14

`label_node()` (*tupa.parse.PassageParser method*), 11

`Labels` (*class in tupa.labels*), 19

`lang_specific` (*tupa.model.ParameterDefinition attribute*), 25

`LeftEdge` (*tupa.action.Actions attribute*), 14

`LeftRemote` (*tupa.action.Actions attribute*), 14

`Lexeme` (*class in tupa.model\_util*), 30

`load()` (*tupa.labels.Labels method*), 19

`load()` (*tupa.model.Model method*), 24

`load_dict()` (*in module tupa.model\_util*), 27

`load_enum()` (*in module tupa.model\_util*), 27

`load_json()` (*in module tupa.model*), 21

`load_json()` (*in module tupa.model\_util*), 28

`load_labels()` (*tupa.model.Model method*), 24

`load_to_config()` (*tupa.model.ParameterDefinition method*), 25

## M

`main()` (*in module tupa.parse*), 6

`main_generator()` (*in module tupa.parse*), 6

`Model` (*class in tupa.model*), 23

`model` (*tupa.parse.AbstractParser attribute*), 8

## N

`need_label()` (*tupa.oracle.Oracle method*), 34

`Node` (*tupa.action.Actions attribute*), 14

`node_label_param_def()` (*tupa.model.Model method*), 24

`num_tokens` (*tupa.parse.PassageParser attribute*), 11

## O

`Oracle` (*class in tupa.oracle*), 34

## P

`param_defs()` (*tupa.model.Model method*), 24

`ParameterDefinition` (*class in tupa.model*), 24

`parse()` (*tupa.parse.BatchParser method*), 8

`parse()` (*tupa.parse.Parser method*), 9

`parse()` (*tupa.parse.PassageParser method*), 11

`parse_internal()` (*tupa.parse.PassageParser method*), 11

`ParseMode` (*class in tupa.parse*), 8

`Parser` (*class in tupa.parse*), 9

`ParserException`, 10

`PassageParser` (*class in tupa.parse*), 10

`percents_str()` (*in module tupa.parse*), 6

`predict()` (*tupa.parse.PassageParser static method*), 11

`print_config()` (*tupa.parse.Parser method*), 10

`print_scores()` (*in module tupa.parse*), 6

## R

[read\\_passages\(\)](#) (in module *tupa.parse*), 6  
[Reduce](#) (*tupa.action.Actions* attribute), 14  
[remote](#) (*tupa.action.Action* attribute), 13  
[RemoteNode](#) (*tupa.action.Actions* attribute), 15  
[remove\(\)](#) (*tupa.oracle.Oracle* method), 34  
[remove\\_backup\(\)](#) (in module *tupa.model*), 21  
[remove\\_backup\(\)](#) (in module *tupa.model\_util*), 28  
[remove\\_existing\(\)](#) (in module *tupa.model\_util*), 28  
[require\\_init\\_features](#)  
     (*tupa.model.ClassifierProperty* attribute),  
     22  
[restore\(\)](#) (*tupa.model.Model* method), 24  
[RightEdge](#) (*tupa.action.Actions* attribute), 15  
[RightRemote](#) (*tupa.action.Actions* attribute), 15

## S

[save\(\)](#) (*tupa.labels.Labels* method), 19  
[save\(\)](#) (*tupa.model.Model* method), 24  
[save\(\)](#) (*tupa.parse.Parser* method), 10  
[save\\_dict\(\)](#) (in module *tupa.model\_util*), 28  
[save\\_json\(\)](#) (in module *tupa.model*), 21  
[save\\_json\(\)](#) (in module *tupa.model\_util*), 28  
[score\(\)](#) (*tupa.model.Model* method), 24  
[set\\_axis\(\)](#) (*tupa.model.Model* method), 24  
[Shift](#) (*tupa.action.Actions* attribute), 15  
[single\\_to\\_iter\(\)](#) (in module *tupa.parse*), 6  
[size\\_arg](#) (*tupa.model.ParameterDefinition* attribute),  
     25  
[str\(\)](#) (*tupa.oracle.Oracle* method), 34  
[Strings](#) (class in *tupa.model\_util*), 30  
[summary\(\)](#) (*tupa.parse.BatchParser* method), 8  
[Swap](#) (*tupa.action.Actions* attribute), 15

## T

[test](#) (*tupa.parse.ParseMode* attribute), 9  
[time\\_per\\_passage\(\)](#) (*tupa.parse.BatchParser*  
     method), 8  
[to\\_lower\\_case\(\)](#) (in module *tupa.parse*), 7  
[tokens\\_per\\_second\(\)](#) (*tupa.parse.AbstractParser*  
     method), 8  
[train](#) (*tupa.parse.ParseMode* attribute), 9  
[train\(\)](#) (*tupa.parse.Parser* method), 10  
[train\\_test\(\)](#) (in module *tupa.parse*), 7  
[trainable\\_after\\_saving](#)  
     (*tupa.model.ClassifierProperty* attribute),  
     22  
[tupa.action](#) (module), 13  
[tupa.config](#) (module), 17  
[tupa.labels](#) (module), 19  
[tupa.model](#) (module), 21  
[tupa.model\\_util](#) (module), 27  
[tupa.oracle](#) (module), 33

[tupa.parse](#) (module), 5  
[type\\_to\\_id](#) (*tupa.action.Action* attribute), 14

## U

[UnknownDict](#) (class in *tupa.model*), 25  
[UnknownDict](#) (class in *tupa.model\_util*), 30  
[update\\_counts\(\)](#) (*tupa.parse.BatchParser* method),  
     8  
[update\\_only\\_on\\_error](#)  
     (*tupa.model.ClassifierProperty* attribute),  
     22

## V

[verify\(\)](#) (*tupa.parse.PassageParser* method), 11  
[Vocab](#) (class in *tupa.model\_util*), 30