
MLConjug Documentation

Release 3.7.20

SekouDiaoNlp

May 04, 2021

Contents

1	MLCONJUG3	3
1.1	Supported Languages	4
1.2	Features	4
1.3	Academic publications citing mlconjug	4
1.4	Software projects using mlconjug	5
1.5	BibTeX	6
1.6	Credits	6
2	Installation	7
2.1	Stable release	7
2.2	From sources	7
3	Usage	9
4	Package Api Documentation for mlconjug	13
4.1	API Reference for the classes in mlconjug3.mlconjug.py	13
4.2	API Reference for the classes in mlconjug3.PyVerbiste.py	15
5	Contributing	21
5.1	Types of Contributions	21
5.2	Get Started!	22
5.3	Pull Request Guidelines	23
5.4	Tips	23
6	Credits	25
6.1	Development Lead	25
6.2	Contributors	25
7	History	27
7.1	3.7.20 (2021-04-05)	27
7.2	3.7.19 (2021-25-04)	27
7.3	3.7.17 (2021-18-04)	27
7.4	3.7.16 (2021-16-04)	28
7.5	3.7.15 (2021-15-04)	28
7.6	3.7.14 (2021-14-04)	28
7.7	3.7.13 (2020-14-10)	28
7.8	3.7.12 (2020-08-10)	28

7.9	3.7.11 (2020-21-09)	29
7.10	3.7.10 (2020-12-09)	29
7.11	3.7.9 (2020-30-08)	29
7.12	3.7.8 (2020-26-08)	29
7.13	3.7.7 (2020-24-08)	29
7.14	3.7.6 (2020-17-05)	29
7.15	3.7.5 (2020-03-05)	29
7.16	3.7.4 (2020-03-05)	30
7.17	3.7.3 (2020-30-04)	30
7.18	3.7.2 (2020-30-04)	30
7.19	3.7.1 (2020-29-01)	30
7.20	3.6.1 (2019-28-11)	30
7.21	3.6.0 (2019-14-11)	30
7.22	3.5.1 (2019-18-07)	31
7.23	3.4 (2019-29-04)	31
7.24	3.3.2 (2019-06-04)	31
7.25	3.3.1 (2019-02-04)	31
7.26	3.3 (2019-04-03)	31
7.27	3.2.3 (2019-26-02)	32
7.28	3.2.2 (2018-18-11)	32
7.29	3.2.0 (2018-04-11)	32
7.30	3.1.3 (2018-07-10)	32
7.31	3.1.2 (2018-06-27)	32
7.32	3.1.1 (2018-06-26)	32
7.33	3.1.0 (2018-06-24)	32
7.34	3.0.1 (2018-06-22)	33
7.35	2.1.11 (2018-06-21)	33
7.36	2.1.9 (2018-06-21)	33
7.37	2.1.5 (2018-06-15)	34
7.38	2.1.2 (2018-06-15)	34
7.39	2.1.0 (2018-06-15)	34
7.40	2.0.0 (2018-06-14)	34
7.41	1.2.0 (2018-06-12)	34
7.42	1.1.0 (2018-06-11)	35
7.43	1.0.0 (2018-06-10)	35
8	Indices and tables	37
	Python Module Index	39
	Index	41

Contents:

CHAPTER 1

MLCONJUG3

A Python library to conjugate verbs in French, English, Spanish, Italian, Portuguese and Romanian (more soon) using Machine Learning techniques.

Any verb in one of the supported language can be conjugated, as the module contains a Machine Learning model of how the verbs behave.

Even completely new or made-up verbs can be successfully conjugated in this manner.

The supplied pre-trained models are composed of:

- a binary feature extractor,
- a feature selector using Linear Support Vector Classification,
- a classifier using Stochastic Gradient Descent.

MLConjug3 uses scikit-learn to implement the Machine Learning algorithms.

Users of the library can use any compatible classifiers from scikit-learn to modify and retrain the models.

The training data for the french model is based on Verbiste <https://perso.b2b2c.ca/~sarrazip/dev/verbiste.html> .

The training data for English, Spanish, Italian, Portuguese and Romanian was generated using unsupervised learning techniques using the French model as a model to query during the training.

Warning: MLCONJUG3 now only supports Python 3.x as Python 2.x has been deprecated in 2020.

- Free software: MIT license
- Documentation: <https://mlconjug3.readthedocs.io>.

1.1 Supported Languages

- French
- English
- Spanish
- Italian
- Portuguese
- Romanian

1.2 Features

- Easy to use API.
- Includes pre-trained models with 99% + accuracy in predicting conjugation class of unknown verbs.
- Easily train new models or add new languages.
- Easily integrate MLConjug in your own projects.
- Can be used as a command line tool.

1.3 Academic publications citing mlconjug

- Ali Malik and Mike Wu and Vrinda Vasavada and Jinpeng Song and John Mitchell and Noah D. Goodman and Chris Piech.
“Generative Grading Neural Approximate Parsing for Automated Student Feedback”.
Proceedings of the 34th AAAI conference on Artificial Intelligence, 2019.

1.4 Software projects using mlconjug

- [NLP-Suite](#)
NLP Suite is a package of tools designed for non-specialists, for scholars with no knowledge or little knowledge of Natural Language Processing.
- [Gender Bias Visualization](#)
This project offers tools to visualize the gender bias in pre-trained language models to better understand the prejudices in the data.
- [Text Adaptation To Context](#)
This project uses language models to generate text that is well suited to the type of publication.
- [verbecc-svc](#)
Dockerized microservice with REST API for conjugation of any verb in French and Spanish.
- [nvhtml](#)
A tool to Manage and transform HTML documents.
- [twitter-bot](#)
Tweets the words of the French language. Largely inspired by the @botducul (identical lexicon, but code in Python) and the @botsupervnr.
Posts on @botduslip. Stores the position of the last tweeted word in a Redis database.
- [verb-form-helper](#)
This project offers a tool to help learn different verbal forms.
- [NLP Tasks](#)
A collection of common NLP tasks such as dataset parsing and explicit semantic extraction.
- [Facemask Detection](#)
This project offers a model which recognizes covid-19 masks.
- [Bad Excuses for Zoom Abuses](#)
Need an excuse for why you can't show up in your Zoom lectures? Just generate one here!
- [NLP](#)
Repository to store Natural Language Processing models.
- [Virtual Assistant](#)
This is a simple virtual assistant. With it, you can search the Internet, access websites, open programs, and more using just your voice.
This virtual assistant supports the English and Portuguese languages and has many settings that you can adjust to your liking.
- [Bad Advice](#)
This python module responds to yes or no questions. It dishes out its advice at random.
Disclaimer: Do not actually act on this advice ;)
- [Spanish Conjugations Quiz](#)
Python+Flask web app that uses mlconjug to dynamically generate foreign language conjugation questions.
- [Silver Rogue DF](#)
A dwarf-fortress adventure mode-inspired rogue-like Pygame Python3 game.

1.5 BibTeX

If you want to cite mlconjug3 in an academic publication use this citation format:

```
@article{mlconjug3,  
  title={mlconjug3},  
  author={Sekou Diao},  
  journal={GitHub. Note: https://github.com/SekouDiaoNlp/mlconjug3 Cited by},  
  year={2021}  
}
```

1.6 Credits

This package was created with the help of [Verbiste](#) and [scikit-learn](#).

The logo was designed by [Zuur](#).

2.1 Stable release

To install MLConjug3, run this command in your terminal:

```
$ pip install mlconjug3
```

This is the preferred method to install MLConjug, as it will always install the most recent stable release.

If you don't have `pip` installed, this [Python installation guide](#) can guide you through the process.

You can also install `mlconjug3` by using [Anaconda](#) or [Miniconda](#) instead of `pip`.

To install [Anaconda](#) or [Miniconda](#), please follow the installation instructions on their respective websites. After having installed [Anaconda](#) or [Miniconda](#), run these commands in your terminal:

```
$ conda config --add channels conda-forge
$ conda config --set channel_priority strict
$ conda install mlconjug3
```

If you already have [Anaconda](#) or [Miniconda](#) available on your system, just type this in your terminal:

```
$ conda install -c conda-forge mlconjug3
```

Warning: If you intend to install `mlconjug3` on a Apple Macbook with an Apple M1 processor, it is advised that you install `mlconjug3` by using the conda installation method as all dependencies will be pre-compiled.

2.2 From sources

The sources for MLConjug can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/SekouDiaoNlp/mlconjug3
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/SekouDiaoNlp/mlconjug3/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

Note: The default language is French. When called without specifying a language, the library will try to conjugate the verb in French.

To use MLConjug3 from the command line:

```
$ mlconjug3 manger

$ mlconjug3 bring -l en

$ mlconjug3 gallofar --language es

$ mlconjug3 -o, --output (Path of the filename for storing the conjugation tables.)

$ mlconjug3 -s, --subject (The subject format type for the conjugated forms). The
                           values can be 'abbrev' or 'pronoun'. The default value
                           is 'abbrev'.

$ mlconjug3 -h Show the help menu
```

To use MLConjug3 in a project with the provided pre-trained conjugation models:

```
import mlconjug3

# To use mlconjug3 with the default parameters and a pre-trained conjugation model.
default_conjugator = mlconjug3.Conjugator(language='fr')

# Verify that the model works
test1 = default_conjugator.conjugate("manger").conjug_info['Indicatif']['Passé Simple
↪']['1p']
test2 = default_conjugator.conjugate("partir").conjug_info['Indicatif']['Passé Simple
↪']['1p']
test3 = default_conjugator.conjugate("facebooker").conjug_info['Indicatif']['Passé_
↪Simple']['1p']
```

(continues on next page)

(continued from previous page)

```

test4 = default_conjugator.conjugate("astigratir").conjug_info['Indicatif']['Passé_
↳Simple']['1p']
test5 = default_conjugator.conjugate("mythoner").conjug_info['Indicatif']['Passé_
↳Simple']['1p']
print(test1)
print(test2)
print(test3)
print(test4)
print(test5)

# You can now iterate over all conjugated forms of a verb by using the newly added_
↳Verb.iterate() method.
default_conjugator = mlconjug3.Conjugator(language='en')
test_verb = default_conjugator.conjugate("be")
all_conjugated_forms = test_verb.iterate()
print(all_conjugated_forms)

```

To use MLConjug3 in a project and train a new model:

```

# Set a language to train the Conjugator on
lang = 'fr'

# Set a ngram range sliding window for the vectorizer
ngramrange = (2,7)

# Transforms dataset with CountVectorizer. We pass the function extract_verb_features_
↳to the CountVectorizer.
vectorizer = mlconjug3.CountVectorizer(analyzer=partial(mlconjug3.extract_verb_
↳features, lang=lang, ngram_range=ngramrange),
                                     binary=True)

# Feature reduction
feature_reductor = mlconjug3.SelectFromModel(mlconjug3.LinearSVC(penalty="l1", max_
↳iter=12000, dual=False, verbose=0))

# Prediction Classifier
classifier = mlconjug3.SGDClassifier(loss="log", penalty='elasticnet', l1_ratio=0.15,
↳max_iter=4000, alpha=1e-5, random_state=42, verbose=0)

# Initialize Data Set
dataset = mlconjug3.DataSet(mlconjug3.Verbiste(language=lang).verbs)
dataset.construct_dict_conjug()
dataset.split_data(proportion=0.9)

# Initialize Conjugator
model = mlconjug3.Model(vectorizer, feature_reductor, classifier)
conjugator = mlconjug3.Conjugator(lang, model)

#Training and prediction
conjugator.model.train(dataset.train_input, dataset.train_labels)
predicted = conjugator.model.predict(dataset.test_input)

# Assess the performance of the model's predictions
score = len([a == b for a, b in zip(predicted, dataset.test_labels) if a == b]) /
↳len(predicted)
print('The score of the model is {0}'.format(score))

```

(continues on next page)

(continued from previous page)

```
# Verify that the model works
test1 = conjugator.conjugate("manger").conjug_info['Indicatif']['Passé Simple']['1p']
test2 = conjugator.conjugate("partir").conjug_info['Indicatif']['Passé Simple']['1p']
test3 = conjugator.conjugate("facebooker").conjug_info['Indicatif']['Passé Simple'][
↪'1p']
test4 = conjugator.conjugate("astigratir").conjug_info['Indicatif']['Passé Simple'][
↪'1p']
test5 = conjugator.conjugate("mythoner").conjug_info['Indicatif']['Passé Simple']['1p
↪']
print(test1)
print(test2)
print(test3)
print(test4)
print(test5)

# Save trained model
with open('path/to/save/data/trained_model-fr.pickle', 'wb') as file:
    pickle.dump(conjugator.model, file)
```

Package Api Documentation for mlconjug

4.1 API Reference for the classes in mlconjug3.mlconjug.py

MLConjug Main module.

This module declares the main classes the user interacts with.

The module defines the classes needed to interface with Machine Learning models.

`mlconjug3.mlconjug.extract_verb_features` (*verb, lang, ngram_range*)

Custom Vectorizer optimized for extracting verbs features.

The Vectorizer subclasses `sklearn.feature_extraction.text.CountVectorizer` .

As in Indo-European languages verbs are inflected by adding a morphological suffix, the vectorizer extracts verb endings and produces a vector representation of the verb with binary features.

To enhance the results of the feature extration, several other features have been included:

The features are the verb's ending n-grams, starting n-grams, length of the verb, number of vowels, number of consonants and the ratio of vowels over consonants.

Parameters

- **verb** – string. Verb to vectorize.
- **lang** – string. Language to analyze.

- **ngram_range** – tuple. The range of the ngram sliding window.

Returns list. List of the most salient features of the verb for the task of finding it's conjugation's class.

class mlconjug3.mlconjug.**Conjugator** (*language='fr', model=None*)

Bases: object

This is the main class of the project.

The class manages the Verbiste data set and provides an interface with the scikit-learn pipeline.

If no parameters are provided, the default language is set to french and the pre-trained french conjugation pipeline is used.

The class defines the method conjugate(verb, language) which is the main method of the module.

Parameters

- **language** – string. Language of the conjugator. The default language is 'fr' for french.
- **model** – mlconjug3.Model or scikit-learn Pipeline or Classifier implementing the fit() and predict() methods. A user provided pipeline if the user has trained his own pipeline.

conjugate (*verb, subject='abbrev'*)

This is the main method of this class.

It first checks to see if the verb is in Verbiste.

If it is not, and a pre-trained scikit-learn pipeline has been supplied, the method then calls the pipeline to predict the conjugation class of the provided verb.

Returns a Verb object or None.

Parameters

- **verb** – string. Verb to conjugate.
- **subject** – string. Toggles abbreviated or full pronouns. The default value is 'abbrev'. Select 'pronoun' for full pronouns.

Returns Verb object or None.

set_model (*model*)

Assigns the provided pre-trained scikit-learn pipeline to be able to conjugate unknown verbs.

Parameters **model** – scikit-learn Classifier or Pipeline.

class mlconjug3.mlconjug.**DataSet** (*verbs_dict*)

Bases: object

This class holds and manages the data set.

Defines helper methods for managing Machine Learning tasks like constructing a training and testing set.

Parameters **verbs_dict** – A dictionary of verbs and their corresponding conjugation class.

construct_dict_conjug ()

Populates the dictionary containing the conjugation templates.

Populates the lists containing the verbs and their templates.

split_data (*threshold=8, proportion=0.5*)

Splits the data into a training and a testing set.

Parameters

- **threshold** – int. Minimum size of conjugation class to be split.
- **proportion** – float. Proportion of samples in the training set. Must be between 0 and 1.

class mlconjug3.mlconjug.**Model** (*vectorizer=None, feature_selector=None, classifier=None, language=None*)

Bases: object

This class manages the scikit-learn pipeline.

The Pipeline includes a feature vectorizer, a feature selector and a classifier.

If any of the vectorizer, feature selector or classifier is not supplied at instance declaration, the `__init__` method will provide good default values that get more than 92% prediction accuracy.

Parameters

- **vectorizer** – scikit-learn Vectorizer.
- **feature_selector** – scikit-learn Classifier with a `fit_transform()` method
- **classifier** – scikit-learn Classifier with a `predict()` method
- **language** – language of the corpus of verbs to be analyzed.

train (*samples, labels*)

Trains the pipeline on the supplied samples and labels.

Parameters

- **samples** – list. List of verbs.
- **labels** – list. List of verb templates.

predict (*verbs*)

Predicts the conjugation class of the provided list of verbs.

Parameters **verbs** – list. List of verbs.

Returns list. List of predicted conjugation groups.

4.2 API Reference for the classes in mlconjug3.PyVerbiste.py

PyVerbiste.

A Python library for conjugating verbs in French, English, Spanish, Italian, Portuguese and Romanian (more soon).

It contains conjugation data generated by machine learning models using the python library mlconjug3.

More information about mlconjug3 at <https://pypi.org/project/mlconjug3/>

The conjugation data conforms to the XML schema defined by Verbiste.

More information on Verbiste at https://perso.b2b2c.ca/~sarrazip/dev/conjug_manager.html

class mlconjug3.PyVerbiste.**ConjugManager** (*language='default'*)

Bases: object

This is the class handling the mlconjug3 json files.

Parameters **language** – string. | The language of the conjugator. The default value is fr for French. | The allowed values are: fr, en, es, it, pt, ro.

_load_verbs (*verbs_file*)

Load and parses the verbs from the json file.

Parameters **verbs_file** – string or path object. Path to the verbs json file.

_load_conjugations (*conjugations_file*)

Load and parses the conjugations from the json file.

Parameters **conjugations_file** – string or path object. Path to the conjugation json file.

_detect_allowed_endings ()

Detects the allowed endings for verbs in the supported languages.

All the supported languages except for English restrict the form a verb can take.

As English is much more productive and varied in the morphology of its verbs, any word is allowed as a verb.

Returns set. A set containing the allowed endings of verbs in the target language.

is_valid_verb (*verb*)

Checks if the verb is a valid verb in the given language.

English words are always treated as possible verbs.

Verbs in other languages are filtered by their endings.

Parameters **verb** – string. The verb to conjugate.

Returns bool. True if the verb is a valid verb in the language. False otherwise.

get_verb_info (*verb*)

Gets verb information and returns a VerbInfo instance.

Parameters **verb** – string. Verb to conjugate.

Returns VerbInfo object or None.

get_conjug_info (*template*)

Gets conjugation information corresponding to the given template.

Parameters **template** – string. Name of the verb ending pattern.

Returns OrderedDict or None. OrderedDict containing the conjugated suffixes of the template.

class mlconjug3.PyVerbiste.**Verbiste** (*language='default'*)

Bases: *mlconjug3.PyVerbiste.ConjugManager*

This is the class handling the Verbiste xml files.

Parameters language – string. | The language of the conjugator. The default value is fr for French. | The allowed values are: fr, en, es, it, pt, ro.

`_load_verbs` (*verbs_file*)

Load and parses the verbs from the xml file.

Parameters verbs_file – string or path object. Path to the verbs xml file.

static `_parse_verbs` (*file*)

Parses the XML file.

Parameters file – FileObject. XML file containing the verbs.

Returns OrderedDict. An OrderedDict containing the verb and its template for all verbs in the file.

`_load_conjugations` (*conjugations_file*)

Load and parses the conjugations from the xml file.

Parameters conjugations_file – string or path object. Path to the conjugation xml file.

`_parse_conjugations` (*file*)

Parses the XML file.

Parameters file – FileObject. XML file containing the conjugation templates.

Returns OrderedDict. An OrderedDict containing all the conjugation templates in the file.

static `_load_tense` (*tense*)

Load and parses the inflected forms of the tense from xml file.

Parameters tense – list of xml tags containing inflected forms. The list of inflected forms for the current tense being processed.

Returns list. List of inflected forms.

`_detect_allowed_endings` ()

Detects the allowed endings for verbs in the supported languages.

All the supported languages except for English restrict the form a verb can take.

As English is much more productive and varied in the morphology of its verbs, any word is allowed as a verb.

Returns set. A set containing the allowed endings of verbs in the target language.

`get_conjug_info` (*template*)

Gets conjugation information corresponding to the given template.

Parameters template – string. Name of the verb ending pattern.

Returns OrderedDict or None. OrderedDict containing the conjugated suffixes of the template.

`get_verb_info` (*verb*)

Gets verb information and returns a VerbInfo instance.

Parameters verb – string. Verb to conjugate.

Returns VerbInfo object or None.

is_valid_verb (*verb*)

Checks if the verb is a valid verb in the given language.

English words are always treated as possible verbs.

Verbs in other languages are filtered by their endings.

Parameters **verb** – string. The verb to conjugate.

Returns bool. True if the verb is a valid verb in the language. False otherwise.

class mlconjug3.PyVerbiste.**VerbInfo** (*infinitive, root, template*)

Bases: object

This class defines the Verbiste verb information structure.

Parameters

- **infinitive** – string. Infinitive form of the verb.
- **root** – string. Lexical root of the verb.
- **template** – string. Name of the verb ending pattern.

class mlconjug3.PyVerbiste.**Verb** (*verb_info, conjug_info, subject='abbrev', predicted=False*)

Bases: object

This class defines the Verb Object.

Parameters

- **verb_info** – VerbInfo Object.
- **conjug_info** – OrderedDict.
- **subject** – string. Toggles abbreviated or full pronouns. The default value is 'abbrev'. Select 'pronoun' for full pronouns.
- **predicted** – bool. Indicates if the conjugation information was predicted by the model or retrieved from the dataset.

iterate ()

Iterates over all conjugated forms and returns a list of tuples of those conjugated forms.

Returns list. List of conjugated forms.

_load_conjug ()

Populates the inflected forms of the verb.

This is the generic version of this method.

It does not add personal pronouns to the conjugated forms.

This method can handle any new language if the conjugation structure conforms to the Verbiste XML Schema.

conjugate_person (*key, persons_dict, term*)

Creates the conjugated form of the person specified by the key argument. :param key: string. :param persons_dict: OrderedDict :param term: string. :return: None.

class mlconjug3.PyVerbiste.**VerbFr** (*verb_info, conjug_info, subject='abbrev', predicted=False*)

Bases: *mlconjug3.PyVerbiste.Verb*

This class defines the French Verb Object.

`_load_conjug()`

Populates the inflected forms of the verb.
Adds personal pronouns to the inflected verbs.

conjugate_person (*key, persons_dict, term*)

Creates the conjugated form of the person specified by the key argument. :param key: string. :param persons_dict: OrderedDict :param term: string. :return: None.

iterate ()

Iterates over all conjugated forms and returns a list of tuples of those conjugated forms.

Returns list. List of conjugated forms.

class mlconjug3.PyVerbiste.**VerbEn** (*verb_info, conjug_info, subject='abbrev', predicted=False*)

Bases: *mlconjug3.PyVerbiste.Verb*

This class defines the English Verb Object.

`_load_conjug()`

Populates the inflected forms of the verb.
Adds personal pronouns to the inflected verbs.

conjugate_person (*key, persons_dict, term*)

Creates the conjugated form of the person specified by the key argument. :param key: string. :param persons_dict: OrderedDict :param term: string. :return: None.

iterate ()

Iterates over all conjugated forms and returns a list of tuples of those conjugated forms.

Returns list. List of conjugated forms.

class mlconjug3.PyVerbiste.**VerbEs** (*verb_info, conjug_info, subject='abbrev', predicted=False*)

Bases: *mlconjug3.PyVerbiste.Verb*

This class defines the Spanish Verb Object.

`_load_conjug()`

Populates the inflected forms of the verb.
Adds personal pronouns to the inflected verbs.

conjugate_person (*key, persons_dict, term*)

Creates the conjugated form of the person specified by the key argument. :param key: string. :param persons_dict: OrderedDict :param term: string. :return: None.

iterate ()

Iterates over all conjugated forms and returns a list of tuples of those conjugated forms.

Returns list. List of conjugated forms.

class mlconjug3.PyVerbiste.**VerbIt** (*verb_info, conjug_info, subject='abbrev', predicted=False*)

Bases: *mlconjug3.PyVerbiste.Verb*

This class defines the Italian Verb Object.

`_load_conjug()`

Populates the inflected forms of the verb.

Adds personal pronouns to the inflected verbs.

conjugate_person (*key, persons_dict, term*)

Creates the conjugated form of the person specified by the key argument. :param key: string. :param persons_dict: OrderedDict :param term: string. :return: None.

iterate ()

Iterates over all conjugated forms and returns a list of tuples of those conjugated forms.

Returns list. List of conjugated forms.

class mlconjug3.PyVerbiste.**VerbPt** (*verb_info, conjug_info, subject='abbrev', predicted=False*)

Bases: *mlconjug3.PyVerbiste.Verb*

This class defines the Portuguese Verb Object.

_load_conjug ()

Populates the inflected forms of the verb.

Adds personal pronouns to the inflected verbs.

conjugate_person (*key, persons_dict, term*)

Creates the conjugated form of the person specified by the key argument. :param key: string. :param persons_dict: OrderedDict :param term: string. :return: None.

iterate ()

Iterates over all conjugated forms and returns a list of tuples of those conjugated forms.

Returns list. List of conjugated forms.

class mlconjug3.PyVerbiste.**VerbRo** (*verb_info, conjug_info, subject='abbrev', predicted=False*)

Bases: *mlconjug3.PyVerbiste.Verb*

This class defines the Romanian Verb Object.

_load_conjug ()

Populates the inflected forms of the verb.

Adds personal pronouns to the inflected verbs.

conjugate_person (*key, persons_dict, term*)

Creates the conjugated form of the person specified by the key argument. :param key: string. :param persons_dict: OrderedDict :param term: string. :return: None.

iterate ()

Iterates over all conjugated forms and returns a list of tuples of those conjugated forms.

Returns list. List of conjugated forms.

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at <https://github.com/SekouDiaoNlp/mlconjug3/issues>.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

5.1.4 Write Documentation

MLConjug could always use more documentation, whether as part of the official MLConjug docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/SekouDiaoNlp/mlconjug3/issues>.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *mlconjug3* for local development.

1. Fork the *mlconjug3* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/mlconjug3.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv mlconjug3
$ cd mlconjug3/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 mlconjug3 tests
$ python setup.py test or py.test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.3, 3.4, 3.5 and 3.6. Check https://travis-ci.org/SekouDiaoNlp/mlconjug3/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_mlconjug
```


6.1 Development Lead

- SekouDiaoNlp <diao.sekou.nlp@gmail.com> GPG key ID: B51D1046EF63C50B

6.2 Contributors

- The logo was designed by [Zuur](#).

7.1 3.7.20 (2021-04-05)

- Retrained all models with scikit-learn 0.24.2
- The Portuguese model has improved and is now also at 99%+ accuracy.
- Updated documentation.
- Updated dependencies.

7.2 3.7.19 (2021-25-04)

- Unknown verb inference is now faster.
- Added more tests to the test suite.
- Automated setup.py dependency injection.
- Updated documentation.
- Updated dependencies.

7.3 3.7.17 (2021-18-04)

- Added new [GitHub Actions Workflows](#) to automatically build, install and test mlconjug3 on Windows-x64, MacOS-x64 and Linux-x64 for maximum compatibility. The package should build and install even on the newest MacBook with an Apple ARM M1 processor by installing mlconjug3 through `conda` instead of `pip`:

```
$ conda install -c conda-forge mlconjug3
```

- Updated the documentation to make it clearer.
- Updated development dependencies.

7.4 3.7.16 (2021-16-04)

- Added support for pipenv environments.
- Added mlconjug3 to [conda-forge](#).
It is especially useful if you want to install mlconjug3 on a MacBook with an Apple M1 processor.
- Now mlconjug3 can be installed using:

```
$ conda install -c conda-forge mlconjug3
```

- Updated documentation.
- Updated dependencies.

7.5 3.7.15 (2021-15-04)

- Updated documentation.
- Updated dependencies.

7.6 3.7.14 (2021-14-04)

- Updated documentation.
- Retrained all models with scikit-learn 0.24.1
- Updated dependencies.

7.7 3.7.13 (2020-14-10)

- Updated documentation.
- Fixed issue#89.
- Added more examples
- Updated dependencies.

7.8 3.7.12 (2020-08-10)

- Updated documentation.
- Added code highnliting for examples.
- Added more examples
- Updated dependencies.

7.9 3.7.11 (2020-21-09)

- Updated documentation.
- Updated dependencies.

7.10 3.7.10 (2020-12-09)

- Fixed errors in English training corpus.
- Retrained English model.
- Updated dependencies.

7.11 3.7.9 (2020-30-08)

- Added Bibtex entry for easier citation in academic publication.

7.12 3.7.8 (2020-26-08)

- Fixed issue #79: Repeated person keys in English present continuous.
- Now the 'person' key of the conjugated forms dictionary can be consistently accessed by [person] for all moods and tenses for a more consistent API.

7.13 3.7.7 (2020-24-08)

- Fixed issue #65 : Infinitive inserted before some conjugated English verbs.
- Fixed issue #66 : Some spanish verbs were not conjugated correctly.
- Retrained all models with scikit-learn 0.23.2.
- Updated dependencies.
- Optimized code to train and predict faster.

7.14 3.7.6 (2020-17-05)

- Fixed issue #47 and #48 where some English and Spanish verbs were not conjugated correctly.
- Fixed issue #50 dealing with some spurious data for Spanish.
- Updated dependencies.

7.15 3.7.5 (2020-03-05)

- Updated the documentation.

7.16 3.7.4 (2020-03-05)

- Fixed issue #44 where Spanish gerunds were not conjugated properly.
- Updated dependencies.

7.17 3.7.3 (2020-30-04)

- Updated the documentation.

7.18 3.7.2 (2020-30-04)

- Fixed issue with package renaming.
- Fixed bug with Portuguese verbs ending in 'ar'.
- Retrained all models with scikit-learn 0.22.2.

7.19 3.7.1 (2020-29-01)

- Updated the pre-trained models for better accuracy (Now all models have more than 99.9% accuracy) .
- Added new utilities for model training and persistence.
- Now all training and GridSearch results are reproducible from run to run.
- Retrained all models with scikit-learn 0.22.1.
- Corrected mutliple edge cases and enlarged the test suite.

7.20 3.6.1 (2019-28-11)

- Updated the pre-trained models for better accuracy (Now all models have more than 99.9% accuracy) .
- Added new utilities for model training and persistence.
- Now all training and GridSearch results are reproducible from run to run.
- Updated development dependencies.

7.21 3.6.0 (2019-14-11)

- Updated scikit-learn dependency to 0.21.3.
- Updated other dependencies.

7.22 3.5.1 (2019-18-07)

- Fixed bug in issue #80 and #81 reported by @rongybika and @NoelHVincent.
- Added new option '-o' to the CLI allowing to specify output file to save results to json file.
- Use logging instead of print() whenever appropriate.
- Use joblib for model persistence instead.
- Updated Type declarations.
- Added more tests in the test-suite.
- Implemented results_parser to select and train the best performing models.
- Implemented multicore grid search.
- Display prettier output in the CLI.
- Updated scikit-learn dependency.
- Updated other dependencies.

7.23 3.4 (2019-29-04)

- Fixed bug when verbs with no common roots with their conjugated form get their root inserted as a prefix.
- Added the method iterate() to the Verb Class as per @poolebu's feature request.
- Updated Dependencies.

7.24 3.3.2 (2019-06-04)

- Corrected bug with regular english verbs not being properly regulated. Thanks to @vectomon
- Updated Dependencies.

7.25 3.3.1 (2019-02-04)

- Corrected bug when updating dependencies to use scikit-learn v 0.20.2 and higher.
- Updated Dependencies.

7.26 3.3 (2019-04-03)

- Updated Dependencies to use scikit-learn v 0.20.2 and higher.
- Updated the pre-trained models to use scikit-learn v 0.20.2 and higher.

7.27 3.2.3 (2019-26-02)

- Updated Dependencies.
- Fixed bug which prevented the installation of the pre-trained models.

7.28 3.2.2 (2018-18-11)

- Updated Dependencies.

7.29 3.2.0 (2018-04-11)

- Updated Dependencies.

7.30 3.1.3 (2018-07-10)

- Updated Documentation.
- Added support for pipenv.
- Included tests and documentation in the package distribution.

7.31 3.1.2 (2018-06-27)

- Updated [Type annotations](#) to the whole library for PEP-561 compliance.

7.32 3.1.1 (2018-06-26)

- Minor Api enhancement (see [API documentation](#))

7.33 3.1.0 (2018-06-24)

- Updated the conjugation models for Spanish and Portuguese.
- Internal changes to the format of the verbiste data from xml to json for better handling of unicode characters.
- New class ConjugManager to more easily add new languages to mlconjug3.
- Minor Api enhancement (see [API documentation](#))

7.34 3.0.1 (2018-06-22)

- **Updated all provided pre-trained prediction models:**
 - Implemented a new vectorizer extracting more meaningful features.
 - As a result the performance of the models has gone through the roof in all languages.
 - Recall and Precision are intesimally close to 100 %. English being the anly to achieve a perfect score at both Recall and Precision.
- **Major API changes:**
 - I removed the class EndingCustomVectorizer and refactored it’s fonctionnality in a top level function called `extract_verb_features()`
 - The provided new improved model are now being zip compressed before release because the feature space has so much grown that their size made them impractical to distribute with the package.
 - Renamed “Model.model” to “Model.pipeline”
 - Renamed “DataSet.liste_verbes” and “DataSet.liste_templates” to “DataSet.verbs_list” and “DataSet.templates_list” respectively. (Pardon my french ;-)
 - Added the attributes “predicted” and “confidence_score” to the class Verb.
 - The whole package have been typed check. I will soon add mlconjug3’s type stubs to typedshed.

7.35 2.1.11 (2018-06-21)

- **Updated all provided pre-trained prediction models**
 - The French Conjugator has accuracy of about 99.94% in predicting the correct conjugation class of a French verb. This is the baseline as i have been working on it for some time now.
 - The English Conjugator has accuracy of about 99.78% in predicting the correct conjugation class of an English verb. This is one of the biggest improvement since version 2.0.0
 - The Spanish Conjugator has accuracy of about 99.65% in predicting the correct conjugation class of a Spanish verb. It has also seen a sizable improvement since version 2.0.0
 - The Romanian Conjugator has accuracy of about 99.06% in predicting the correct conjugation class of a Romanian verb.This is by far the bigger gain. I modified the vectorizer to better take into account the morphological features or romanian verbs. (the previous score was about 86%, so it wil be nice for our romanian friends to have a trusted conjugator)
 - The Portuguese Conjugator has accuracy of about 96.73% in predicting the correct conjugation class of a Portuguese verb.
 - The Italian Conjugator has accuracy of about 94.05% in predicting the correct conjugation class of a Italian verb.

7.36 2.1.9 (2018-06-21)

- **Now the Conjugator adds additional information to the Verb object returned.**
 - If the verb under consideration is already in Verbiste, the conjugation for the verb is retrieved directly from memory.

- If the verb under consideration is unknown in Verbiste, the Conjugator class now sets the boolean attribute 'predicted' and the float attribute confidence score to the instance of the Verb object the Conjugator.conjugate(verb) returns.
- Added [Type annotations](#) to the whole library for robustness and ease of scaling-out.
- The performance of the English and Romanian Models have improved significantly lately. I guess in a few more iteration they will be on par with the French Model which is the best performing at the moment as i have been tuning its parameters for a caouple of year now. Not so much with the other languages, but if you update regularly you will see nice improvents in the 2.2 release.
- Enhanced the localization of the program.
- Now the user interface of mlconjug3 is avalaible in French, Spanish, Italian, Portuguese and Romanian, in addition to English.
- [All the documentation of the project](#) have been translated in the supported languages.

7.37 2.1.5 (2018-06-15)

- Added localization.
- Now the user interface of mlconjug3 is avalaible in French, Spanish, Italian, Portuguese and Romanian, in addition to English.

7.38 2.1.2 (2018-06-15)

- Added invalid verb detection.

7.39 2.1.0 (2018-06-15)

- Updated all language models for compatibility with scikit-learn 0.19.1.

7.40 2.0.0 (2018-06-14)

- Includes English conjugation model.
- Includes Spanish conjugation model.
- Includes Italian conjugation model.
- Includes Portuguese conjugation model.
- Includes Romanian conjugation model.

7.41 1.2.0 (2018-06-12)

- Refactored the API. Now a Single class Conjugator is needed to interface with the module.
- Includes improved french conjugation model.
- Added support for multiple languages.

7.42 1.1.0 (2018-06-11)

- Refactored the API. Now a Single class Conjugator is needed to interface with the module.
- Includes improved french conjugation model.

7.43 1.0.0 (2018-06-10)

- First release on PyPI.

CHAPTER 8

Indices and tables

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

m

`mlconjug3.mlconjug`, [13](#)
`mlconjug3.PyVerbiste`, [15](#)

Symbols

- `_detect_allowed_endings()` (*mlconjug3.PyVerbiste.ConjugManager method*), 16
- `_detect_allowed_endings()` (*mlconjug3.PyVerbiste.Verbiste method*), 17
- `_load_conjug()` (*mlconjug3.PyVerbiste.Verb method*), 18
- `_load_conjug()` (*mlconjug3.PyVerbiste.VerbEn method*), 19
- `_load_conjug()` (*mlconjug3.PyVerbiste.VerbEs method*), 19
- `_load_conjug()` (*mlconjug3.PyVerbiste.VerbFr method*), 18
- `_load_conjug()` (*mlconjug3.PyVerbiste.VerbIt method*), 19
- `_load_conjug()` (*mlconjug3.PyVerbiste.VerbPt method*), 20
- `_load_conjug()` (*mlconjug3.PyVerbiste.VerbRo method*), 20
- `_load_conjugations()` (*mlconjug3.PyVerbiste.ConjugManager method*), 16
- `_load_conjugations()` (*mlconjug3.PyVerbiste.Verbiste method*), 17
- `_load_tense()` (*mlconjug3.PyVerbiste.Verbiste static method*), 17
- `_load_verbs()` (*mlconjug3.PyVerbiste.ConjugManager method*), 16
- `_load_verbs()` (*mlconjug3.PyVerbiste.Verbiste method*), 17
- `_parse_conjugations()` (*mlconjug3.PyVerbiste.Verbiste method*), 17
- `_parse_verbs()` (*mlconjug3.PyVerbiste.Verbiste static method*), 17
- C**
- `conjugate()` (*mlconjug3.mlconjug.Conjugator method*), 14
- `conjugate_person()` (*mlconjug3.PyVerbiste.Verb method*), 18
- `conjugate_person()` (*mlconjug3.PyVerbiste.VerbEn method*), 19
- `conjugate_person()` (*mlconjug3.PyVerbiste.VerbEs method*), 19
- `conjugate_person()` (*mlconjug3.PyVerbiste.VerbFr method*), 19
- `conjugate_person()` (*mlconjug3.PyVerbiste.VerbIt method*), 20
- `conjugate_person()` (*mlconjug3.PyVerbiste.VerbPt method*), 20
- `conjugate_person()` (*mlconjug3.PyVerbiste.VerbRo method*), 20
- Conjugator (*class in mlconjug3.mlconjug*), 14
- ConjugManager (*class in mlconjug3.PyVerbiste*), 16
- `construct_dict_conjug()` (*mlconjug3.mlconjug.DataSet method*), 14
- D**
- DataSet (*class in mlconjug3.mlconjug*), 14
- E**
- `extract_verb_features()` (*in module mlconjug3.mlconjug*), 13
- G**
- `get_conjug_info()` (*mlconjug3.PyVerbiste.ConjugManager method*), 16
- `get_conjug_info()` (*mlconjug3.PyVerbiste.Verbiste method*), 17
- `get_verb_info()` (*mlconjug3.PyVerbiste.ConjugManager method*), 16
- `get_verb_info()` (*mlconjug3.PyVerbiste.Verbiste method*), 17

I

`is_valid_verb()` (*mlconjug3.PyVerbiste.ConjugManager method*), 16

`is_valid_verb()` (*mlconjug3.PyVerbiste.Verbiste method*), 17

`iterate()` (*mlconjug3.PyVerbiste.Verb method*), 18

`iterate()` (*mlconjug3.PyVerbiste.VerbEn method*), 19

`iterate()` (*mlconjug3.PyVerbiste.VerbEs method*), 19

`iterate()` (*mlconjug3.PyVerbiste.VerbFr method*), 19

`iterate()` (*mlconjug3.PyVerbiste.VerbIt method*), 20

`iterate()` (*mlconjug3.PyVerbiste.VerbPt method*), 20

`iterate()` (*mlconjug3.PyVerbiste.VerbRo method*), 20

M

`mlconjug3.mlconjug` (*module*), 13

`mlconjug3.PyVerbiste` (*module*), 15

`Model` (*class in mlconjug3.mlconjug*), 15

P

`predict()` (*mlconjug3.mlconjug.Model method*), 15

S

`set_model()` (*mlconjug3.mlconjug.Conjugator method*), 14

`split_data()` (*mlconjug3.mlconjug.DataSet method*), 15

T

`train()` (*mlconjug3.mlconjug.Model method*), 15

V

`Verb` (*class in mlconjug3.PyVerbiste*), 18

`VerbEn` (*class in mlconjug3.PyVerbiste*), 19

`VerbEs` (*class in mlconjug3.PyVerbiste*), 19

`VerbFr` (*class in mlconjug3.PyVerbiste*), 18

`VerbInfo` (*class in mlconjug3.PyVerbiste*), 18

`Verbiste` (*class in mlconjug3.PyVerbiste*), 16

`VerbIt` (*class in mlconjug3.PyVerbiste*), 19

`VerbPt` (*class in mlconjug3.PyVerbiste*), 20

`VerbRo` (*class in mlconjug3.PyVerbiste*), 20