MLConjug Documentation

Release 3.4.0

SekouD

29 apr 2019

Indice

Contenuto:

CAPITOLO 1

mlconjug

Una libreria Python per coniugare verbi in francese, inglese, spagnolo, italiano, portoghese e rumeno (più presto) usando tecniche di Machine Learning.
Qualsiasi verbo in una delle lingue supportate può essere coniugato, in quanto il modulo contiene un modello di Machine Learning su come si comportano i verbi.
Anche verbi completamente nuovi o inventati possono essere coniugati con successo in questo modo.
I modelli pre-formati forniti sono composti da:

- un estrattore di feature binario
- un selettore di funzionalità che utilizza la classificazione vettoriale di supporto lineare
- un classificatore che utilizza la discesa gradiente stocastica»

MLConjug usa scikit-imparare per implementare gli algoritmi di Machine Learning.
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 .

I dati di addestramento per inglese, spagnolo, italiano, portoghese e rumeno sono stati generati utilizzando tecniche di apprendimento senza supervisione utilizzando il modello francese come modello da interrogare durante la formazione.

- Software gratuito: licenza MIT
- Documentazione: https://mlconjug.readthedocs.io.

1.1 Lingue supportate

- Francese
- Inglese
- Spagnolo
- Italiano
- Portoghese
- Rumeno

1.2 Caratteristiche

- API facile da usare»
- Include modelli pre-addestrati con un'accuratezza del 99% + nel predire la classe di coniugazione di verbi sconosciuti.
- Allena facilmente nuovi modelli o aggiungi nuove lingue.
- Integrate facilmente MLConjug nei vostri progetti.
- Può essere usato come uno strumento da riga di comando.

1.3 Crediti

Questo pacchetto è stato creato con l'aiuto di Verbiste e scikit-learn.

Il logo è stato progettato da Zuur.

CAPITOLO 2

Installazione

2.1 Rilascio stabile

Per installare MLConjug, esegui questo comando nel tuo terminale:

```
$ pip install mlconjug
```

Questo è il metodo preferito per installare MLConjug, poiché installerà sempre la versione stabile più recente.

Se non hai installato‘ pip‘_, questa Guida all’installazione di Python può guidarti attraverso il processo.

2.2 Dalle fonti

Le fonti per MLConjug possono essere scaricate dal‘ Github repo‘_.

Puoi clonare il repository pubblico:

```
$ git clone git://github.com/SekouD/mlconjug
```

Oppure scarica il‘ tarball‘_:

```
$ curl  -OL https://github.com/SekouD/mlconjug/tarball/master
```

Una volta che hai una copia del codice sorgente, puoi installarlo con:

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

CAPITOLO 3

Utilizzo

Nota: La lingua predefinita è il francese: quando viene chiamato senza specificare una lingua, la libreria proverà a coniugare il verbo in francese»

Utilizzare MLConjug in un progetto con i modelli di coniugazione pre-addestrati forniti:

```
import mlconjug

# To use mlconjug with the default parameters and a pre-trained conjugation model.
default_conjugator = mlconjug.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']
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 = mlconjug.Conjugator(language='en')
test_verb = default_conjugator.conjugate("be")
all_conjugated_forms = test_verb.iterate()
print(all_conjugated_forms)
```

Utilizzare MLConjug in un progetto e formare un nuovo modello

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

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

# Transforms dataset with CountVectorizer. We pass the function extract_verb_features␣
↪to the CountVectorizer.
vectorizer = mlconjug.CountVectorizer(analyzer=partial(mlconjug.extract_verb_features,
↪ lang=lang, ngram_range=ngrange),
                              binary=True)

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

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

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

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

# 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)
```

Per utilizzare MLConjug dalla riga di comando

```
$ mlconjug manger

$ mlconjug bring -l en

$ mlconjug gallofar --language es
```

CAPITOLO 4

Documentazione pacchetto API per mlconjug

4.1 Riferimento API per le classi in mlconjug.mlconjug.py

Modulo MLConjug Main.

Questo modulo dichiara le classi principali con cui l'utente interagisce.

Il modulo definisce le classi necessarie per interfacciarsi con i modelli Machine Learning.

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

Vectorizer personalizzato ottimizzato per l'estrazione delle funzionalità dei verbi.
Le sottoclassi di Vectorizer 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.

Per migliorare i risultati dell'estrazione delle funzionalità, sono state incluse molte altre funzionalità:

Le caratteristiche sono i n-grammi finali del verbo, che iniziano con n-grammi, la lunghezza del verbo, il numero di vocali, il numero di consonanti e il rapporto delle vocali rispetto alle consonanti.

Parametri

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

- **ngram_range** – tuple. La gamma della finestra scorrevole di ngram.

Ritorna elenco. Elenco delle caratteristiche più salienti del verbo per il compito di trovare la classe di coniugazione.

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

Questa è la classe principale del progetto.

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.

La classe definisce il metodo conjugate (verb, language) che è il metodo principale del modulo.

Parametri

- **language** – string. Lingua del coniugatore. La lingua predefinita è» fr «per il francese.
- **model** – mlconjug.Model o scikit-learn Pipeline o Classifier che implementano i metodi fit () e prediction (). Una pipeline fornita dall'utente se l'utente ha addestrato la propria pipeline.

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

Questo è il metodo principale di questa classe.

Prima controlla se il verbo è 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.

Restituisce un oggetto Verb o None.

Parametri

- **verb** – string. Verb to conjugate»
- **subject** – string Attiva i pronomi abbreviati o pieni Il valore predefinito è» abbrev «Seleziona» pronome «per i pronomi completi.

Ritorna Oggetto Verb o None.

set_model(*model*)

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

Parametri **model** – Scikit-learn Classifier o Pipeline.

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

Questa classe contiene e gestisce il set di dati.

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

Parametri **verbs_dict** – Un dizionario di verbi e la loro corrispondente classe di coniugazione.

construct_dict_conjug()

Compila il dizionario contenente i modelli di coniugazione.
Compila gli elenchi contenenti i verbi e i loro modelli.

split_data(*threshold=8*, *proportion=0.5*)
Divide i dati in una formazione e un set di test.

Parametri

- **threshold** – int. Dimensione minima della classe di coniugazione da dividere»
- **proportion** – float. Proporzione di campioni nel set di allenamento. Deve essere compresa tra 0 e 1.

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

Basi: object

This class manages the scikit-learn pipeline.
La pipeline include un vettore di funzionalità, un selettore di funzionalità e un classificatore.
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.

Parametri

- **vectorizer** – Scikit-learn Vectorizer.
- **feature_selector** – Scikit-learn Classifier con un metodo fit_transform ()
- **classifier** – Scikit-learn Classifier con un metodo di previsione ()
- **language** – linguaggio del corpus di verbi da analizzare.

train(*samples*, *labels*)
Trains the pipeline on the supplied samples and labels.

Parametri

- **samples** – sgstr «Elenco dei verbi.
- **labels** – sgstr «Elenco di modelli di verbi.

predict(*verbs*)
Predice la classe di coniugazione dell'elenco di verbi fornito.

Parametri **verbs** – sgstr «Elenco dei verbi.

Ritorna sgstr «Elenco dei gruppi di coniugazione previsti.

4.2 Riferimento API per le classi in mlconjug.PyVerbiste.py

PyVerbiste.

Una libreria Python per coniugare i verbi in francese, inglese, spagnolo, italiano, portoghese e rumeno (più presto).
Contiene dati di coniugazione generati da modelli di apprendimento automatico che utilizzano la libreria python mlconjug.
Maggiori informazioni su mlconjug all'indirizzo https://pypi.org/project/mlconjug/

I dati di coniugazione sono conformi allo schema XML definito da Verbiste.
Maggiori informazioni su Verbiste su https://perso.b2b2c.ca/~sarrazip/dev/conjug_manager.html

class mlconjug.PyVerbiste.**ConjugManager**(*language='default'*)
This is the class handling the mlconjug json files.

Parametri **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.

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

_load_conjugations(*conjugations_file*)
Load and parses the conjugations from the xml file.

Parametri **conjugations_file** – oggetto stringa o percorso Percorso per il file xml di coniugazione.

_detect_allowed_endings()

Rileva le terminazioni consentite per i verbi nelle lingue supportate.
Tutte le lingue supportate tranne l'inglese limitano il formato che un verbo può assumere.
Poiché l'inglese è molto più produttivo e vario nella morfologia dei suoi verbi, ogni parola è consentita come un verbo.

Ritorna set. Un set contenente le terminazioni consentite di verbi nella lingua di destinazione.

is_valid_verb(*verb*)

Controlla se il verbo è un verbo valido nella lingua specificata.
Le parole inglesi sono sempre trattate come possibili verbi.
I verbi in altre lingue sono filtrati dai loro finali.

Parametri **verb** – stringa, il verbo coniugare.

Ritorna bool, vero se il verbo è un verbo valido nella lingua, altrimenti falso.

get_verb_info(*verb*)
Ottiene informazioni sui verbi e restituisce un'istanza VerbInfo.

Parametri **verb** – string. Verb to conjugate»

Ritorna Oggetto VerbInfo o Nessuno.

get_conjug_info(*template*)
Ottiene informazioni di coniugazione corrispondenti al modello specificato.

Parametri **template** – string Nome del modello di fine del verbo.

Ritorna OrderedDict o None. OrderedDict contenente i suffissi coniugati del modello.

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

Basi: *mlconjug.PyVerbiste.ConjugManager*

Questa è la classe che gestisce i file xml di Verbiste.

Parametri **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.

Parametri **verbs_file** – oggetto stringa o percorso. Percorso del file xml verbi.

_parse_verbs(*file*)
Parses the XML file.

Parametri **file** – FileObject. File XML contenente i verbi.

Ritorna OrderedDict. Un OrderedDict contenente il verbo e il suo modello per tutti i verbi nel file.

_load_conjugations(*conjugations_file*)
Load and parses the conjugations from the xml file.

Parametri **conjugations_file** – oggetto stringa o percorso Percorso per il file xml di coniugazione.

_parse_conjugations(*file*)
Parses the XML file.

Parametri **file** – FileObject. File XML contenente i modelli di coniugazione.

Ritorna OrderedDict. Un OrderedDict contenente tutti i modelli di coniugazione nel file.

_load_tense(*tense*)
Carica e analizza le forme flesse del tempo dal file xml.

Parametri **tense** – elenco di tag xml contenenti moduli flessi. L'elenco delle forme flesse per il tempo corrente in elaborazione.

Ritorna list. List of inflected forms.

_detect_allowed_endings()

Rileva le terminazioni consentite per i verbi nelle lingue supportate.
Tutte le lingue supportate tranne l'inglese limitano il formato che un verbo può assumere.
Poiché l'inglese è molto più produttivo e vario nella morfologia dei suoi verbi, ogni parola è consentita come un verbo.

Ritorna set. Un set contenente le terminazioni consentite di verbi nella lingua di destinazione.

get_conjug_info(*template*)
Ottiene informazioni di coniugazione corrispondenti al modello specificato.

Parametri **template** – string Nome del modello di fine del verbo.

Ritorna OrderedDict o None. OrderedDict contenente i suffissi coniugati del modello.

get_verb_info(*verb*)
Ottiene informazioni sui verbi e restituisce un'istanza VerbInfo.

Parametri **verb** – string. Verb to conjugate»

Ritorna Oggetto VerbInfo o Nessuno.

is_valid_verb(*verb*)

Controlla se il verbo è un verbo valido nella lingua specificata.

Le parole inglesi sono sempre trattate come possibili verbi.
I verbi in altre lingue sono filtrati dai loro finali.

Parametri **verb** – stringa, il verbo coniugare.

Ritorna bool, vero se il verbo è un verbo valido nella lingua, altrimenti falso.

class mlconjug.PyVerbiste.**VerbInfo**(*infinitive*, *root*, *template*)
Questa classe definisce la struttura delle informazioni sui verbi del Verbese.

Parametri

- **infinitive** – stringa, forma infinita del verbo»
- **root** – stringa, radice lessicale del verbo»
- **template** – string Nome del modello di fine del verbo.

class mlconjug.PyVerbiste.**Verb**(*verb_info*, *conjug_info*, *subject='abbrev'*, *predicted=False*)
This class defines the Verb Object. TODO: Make the conjugated forms iterable by implementing the iterator protocol.

Parametri

- **verb_info** – Oggetto VerbInfo.
- **conjug_info** – OrderedDict.
- **subject** – string Attiva i pronomi abbreviati o pieni Il valore predefinito è» abbrev «Seleziona» pronome «per i pronomi completi.
- **predicted** – bool. Indica se le informazioni di coniugazione sono state previste dal modello o recuperate dal set di dati.

iterate()
Iterates over all conjugated forms and returns a list of tuples of those conjugated forms. :return:

_load_conjug()

Compila le forme flesse del verbo.
Questa è la versione generica di questo metodo.
Non aggiunge pronomi personali alle forme coniugate.
Questo metodo può gestire qualsiasi nuovo linguaggio se la struttura di coniugazione è conforme allo schema XML di Verbiste.

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

Basi: *mlconjug.PyVerbiste.Verb*

Questa classe definisce l'oggetto verbo francese.

_load_conjug()

Compila le forme flesse del verbo.
Aggiunge pronomi personali ai verbi incurvati.

iterate()
Iterates over all conjugated forms and returns a list of tuples of those conjugated forms. :return:

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

Basi: *mlconjug.PyVerbiste.Verb*

Questa classe definisce l'oggetto verbo inglese.

_load_conjug()

Compila le forme flesse del verbo.
Aggiunge pronomi personali ai verbi incurvati.

iterate()
Iterates over all conjugated forms and returns a list of tuples of those conjugated forms. :return:

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

Basi: *mlconjug.PyVerbiste.Verb*

Questa classe definisce l'oggetto verbo spagnolo.

_load_conjug()

Compila le forme flesse del verbo.
Aggiunge pronomi personali ai verbi incurvati.

iterate()
Iterates over all conjugated forms and returns a list of tuples of those conjugated forms. :return:

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

Basi: *mlconjug.PyVerbiste.Verb*

Questa classe definisce l'oggetto Verb italiano.

_load_conjug()

Compila le forme flesse del verbo.
Aggiunge pronomi personali ai verbi incurvati.

iterate()
Iterates over all conjugated forms and returns a list of tuples of those conjugated forms. :return:

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

Basi: *mlconjug.PyVerbiste.Verb*

Questa classe definisce l'oggetto Verb Portoghese.

_load_conjug()

Compila le forme flesse del verbo.
Aggiunge pronomi personali ai verbi incurvati.

iterate()
Iterates over all conjugated forms and returns a list of tuples of those conjugated forms. :return:

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

Basi: *mlconjug.PyVerbiste.Verb*

Questa classe definisce l'oggetto Verb Rumeno.

iterate()
Iterates over all conjugated forms and returns a list of tuples of those conjugated forms. :return:

_load_conjug()

Compila le forme flesse del verbo.
Aggiunge pronomi personali ai verbi incurvati.

CAPITOLO 5

Contribuire

I contributi sono ben accetti e sono molto apprezzati! Ogni piccolo aiuto e il credito verrà sempre dato»

Puoi contribuire in molti modi:

5.1 Tipi di contributi

5.1.1 Segnala bug

Segnala bug a https://github.com/SekouD/mlconjug/issues.

Se stai segnalando un bug, includi:

- Nome e versione del tuo sistema operativo.
- Qualsiasi dettaglio sulla configurazione locale che potrebbe essere utile nella risoluzione dei problemi.
- Passaggi dettagliati per riprodurre il bug.

5.1.2 Bug corretti

Cerca tra i problemi di GitHub per i bug. Qualsiasi cosa taggata con » bug «e » help wanted «è aperta a chiunque voglia implementarla.

5.1.3 Implementa funzionalità

Cerca tra i problemi di GitHub per le funzionalità. Qualsiasi cosa taggata con » enhancement «e » help wanted «è aperta a chiunque voglia implementarla.

5.1.4 Scrivi documentazione

MLConjug potrebbe sempre usare più documentazione, sia come parte dei documenti ufficiali di MLConjug, in docstrings, o anche sul Web in post di blog, articoli e simili.

5.1.5 Invia feedback

Il modo migliore per inviare feedback è presentare un problema su https://github.com/SekouD/mlconjug/issues.

Se stai proponendo una funzione:

- Spiega in dettaglio come funzionerebbe.
- Tieni l'ambito il più stretto possibile, per renderlo più facile da implementare.
- Ricorda che questo è un progetto guidato dai volontari e che i contributi sono i benvenuti :)

5.2 Iniziare!

Pronto a contribuire? Ecco come impostare‘ mlconjug‘ per lo sviluppo locale.

1. Prepara il repo “mlconjug‘ su GitHub.
2. Clona la tua forcella localmente

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

3. Installa la tua copia locale in un virtualenv. Supponendo che hai virtualenvwrapper installato, questo è il modo in cui imposti la tua forcella per lo sviluppo locale

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

4. Crea un ramo per lo sviluppo locale

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

   Ora puoi apportare le modifiche localmente.

5. Quando hai finito di fare modifiche, controlla che le tue modifiche superino flake8 e i test, incluso testare altre versioni di Python con tox

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

   Per ottenere flake8 e tox, basta installarli nel tuo virtualenv.

6. Confida le tue modifiche e invia il tuo ramo a GitHub

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

7. Invia una richiesta di pull tramite il sito Web GitHub.

5.3 Pull Request Guidelines

Prima di inviare una richiesta di pull, controlla che soddisfi le seguenti linee guida:

1. La richiesta di pull dovrebbe includere test.
2. Se la richiesta pull aggiunge funzionalità, i documenti dovrebbero essere aggiornati. Inserisci la tua nuova funzionalità in una funzione con una docstring e aggiungi la funzione all'elenco 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/SekouD/mlconjug/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Suggerimenti

Per eseguire un sottoinsieme di test

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

CAPITOLO 6

Crediti

6.1 Lead di sviluppo

- SekouD <sekoud.python@gmail.com> GPG key ID: B51D1046EF63C50B

6.2 Contributori

- Il logo è stato progettato da Zuur.

CAPITOLO 7

Storia

7.1 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.

7.2 3.3.2 (2019-06-04)

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

7.3 3.3.1 (2019-02-04)

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

7.4 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.

7.5 3.2.3 (2019-26-02)

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

7.6 3.2.2 (2018-18-11)

- Updated Dependencies.

7.7 3.2.0 (2018-04-11)

- Updated Dependencies.

7.8 3.1.3 (2018-07-10)

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

7.9 3.1.2 (2018-06-27)

- Aggiornato Scrivi annotazioni sull'intera libreria per la conformità PEP-561.

7.10 3.1.1 (2018-06-26)

- Miglioramento di Api minori (vedere Documentazione API)

7.11 3.1.0 (2018-06-24)

- Aggiornati i modelli di coniugazione per lo spagnolo e il portoghese.
- Modifiche interne al formato dei dati verbiste da xml a json per una migliore gestione dei caratteri Unicode.
- ConjugManager di nuova classe per aggiungere più facilmente nuove lingue a mlconjug.
- Miglioramento di Api minori (vedere Documentazione API)

7.12 3.0.1 (2018-06-22)

- **Aggiornati tutti i modelli di previsione pre-formati forniti:**
  - Implementato un nuovo vectrorizer che estrae funzionalità più significative.
  - Di conseguenza, le prestazioni dei modelli hanno attraversato il tetto in tutte le lingue.
  - Recall e Precision sono intimamente vicini al 100%. L'inglese è l'unico a ottenere un punteggio perfetto sia in Recall che in Precision.
- **Principali modifiche API:**
  - Ho rimosso la classe EndingCustomVectorizer e ho rifattorizzato la sua funzionalità in una funzione di livello superiore chiamata extract_verb_features ()
  - Il nuovo modello migliorato fornito viene ora compresso tramite zip prima del rilascio perché lo spazio delle funzioni è cresciuto così tanto che le loro dimensioni li hanno resi poco pratici da distribuire con il pacchetto.
  - Rinominato «Model.model» in «Model.pipeline»
  - Rinominati «DataSet.liste_verbes» e «DataSet.liste_templates» rispettivamente in «DataSet.verbs_list» e «DataSet.templates_list». (Perdona il mio francese ;-))
  - Aggiunti gli attributi «predetto» e «confidence_score» alla classe Verb.
  - L'intero pacchetto è stato controllato a macchina. Presto aggiungerò gli stub del tipo di mlconjug a typeshed.

7.13 2.1.11 (2018-06-21)

- **Aggiornato tutti i modelli di previsione pre-formati forniti**
  - Il coniugatore francese ha una precisione di circa il 99,94% nel predire la corretta classe di coniugazione di un verbo francese. Questa è la linea di base, come ho lavorato su di esso per qualche tempo.
  - Il coniugatore inglese ha una precisione di circa il 99,78% nel predire la corretta classe di coniugazione di un verbo inglese. Questo è uno dei maggiori miglioramenti dalla versione 2.0.0
  - Il coniugatore spagnolo ha una precisione di circa il 99,65% nel predire la corretta classe di coniugazione di un verbo spagnolo. Ha anche visto un notevole miglioramento dalla versione 2.0.0
  - Il coniugatore rumeno ha una precisione di circa il 99,06% nel predire la corretta classe di coniugazione di un verbo rumeno. Questo è di gran lunga il guadagno maggiore. Ho modificato il vettore per meglio tenere conto delle caratteristiche morfologiche o dei verbi rumeni. (il punteggio precedente era circa l'86%, quindi sarà bello per i nostri amici rumeni avere un coniuge affidabile)
  - Il coniugatore portoghese ha una precisione di circa il 96,73% nel predire la corretta classe di coniugazione di un verbo portoghese.
  - Il coniugatore italiano ha una precisione di circa il 94,05% nel predire la corretta classe di coniugazione di un verbo italiano.

7.14 2.1.9 (2018-06-21)

- **Ora il coniugatore aggiunge ulteriori informazioni all'oggetto Verb restituito.**

  - Se il verbo preso in considerazione è già in Verbiste, la coniugazione per il verbo viene recuperata direttamente dalla memoria.
  - Se il verbo in considerazione è sconosciuto in Verbiste, la classe Coniugatore imposta ora l'attributo booleano "predetto" e il punteggio di confidenza dell'attributo float sull'istanza dell'oggetto Verb che restituisce il coniugatore.conjugato (verbo).

- Aggiunte le annotazioni di tipo su tutta la libreria per robustezza e facilità di ridimensionamento.
- Le prestazioni dei modelli engish e rumeni sono migliorate significativamente negli ultimi tempi. Immagino che in qualche altra versione saranno alla pari con il modello francese che al momento è il migliore in quanto ho messo a punto i suoi parametri per un anno intero. Non così tanto con le altre lingue, ma se aggiorni regolarmente vedrai dei bei improvvisi nella versione 2.2.
- Migliorata la localizzazione del programma.
- Ora l'interfaccia utente di mlconjug è disponibile in francese, spagnolo, italiano, portoghese e rumeno, oltre all'inglese.
- Tutta la documentazione del progetto è stata tradotta nelle lingue supportate.

7.15 2.1.5 (2018-06-15)

- Aggiunta localizzazione.
- Ora l'interfaccia utente di mlconjug è disponibile in francese, spagnolo, italiano, portoghese e rumeno, oltre all'inglese.

7.16 2.1.2 (2018-06-15)

- Aggiunto riconoscimento verbo non valido.

7.17 2.1.0 (2018-06-15)

- Aggiornamento di tutti i modelli linguistici per compatibilità con scikit-learn 0.19.1.

7.18 2.0.0 (2018-06-14)

- Include il modello di coniugazione inglese.
- Include il modello di coniugazione spagnolo.
- Include il modello di coniugazione italiano.
- Include il modello di coniugazione portoghese.
- Include il modello di coniugazione rumeno.

7.19 1.2.0 (2018-06-12)

- Rifattorizzato l'API. Ora è necessario un coniugatore di classe singola per interfacciarsi con il modulo.
- Include un modello di coniugazione francese migliorato.
- Aggiunto supporto per più lingue.

7.20 1.1.0 (2018-06-11)

- Rifattorizzato l'API. Ora è necessario un coniugatore di classe singola per interfacciarsi con il modulo.
- Include un modello di coniugazione francese migliorato.

7.21 1.0.0 (2018-06-10)

- Prima versione su PyPI.

CAPITOLO 8

Indici e tabelle

- genindex
- modindex
- search

Indice del modulo Python

m

Indice

Simboli

C

D

E

G

I

M

P

S

T

V