MLConjug Documentation

Version 3.4.0

SekouD

avr. 29, 2019

Table des matières

Contenu :

CHAPITRE 1

mlconjug

Une librairie Python pour conjuguer les verbes en français, anglais, espagnol, italien, portugais et roumain (plus bientôt) en utilisant des techniques d'apprentissage automatique autrement appelées Machine Learning.

N'importe quel verbe dans l'une des langues prises en charge peut être conjugué, car le module contient un modèle d'apprentissage automatique du comportement des verbes.

Même des verbes complètement nouveaux ou inventés peuvent être conjugués avec succès de cette manière.

Les modèles pré-entraînés fournis sont composés de :

- un extracteur de traits binaire (caractéristiques saillantes des verbes),
- un sélecteur de caractéristiques utilisant la classification linéaire vectorisée de support (Linear Support Vector Classification),
- un classificateur utilisant la descente de gradient stochastique (Stochastic Gradient Descent).

MLConjug utilise scikit-learn pour implémenter les algorithmes de Machine Learning.

Les utilisateurs de la librairie peuvent utiliser n'importe quel classificateur compatible avec l'API scikit-learn pour modifier et entraîner le modèle.

Les données d'entraînement pour le français sont basées sur Verbiste
https://perso.b2b2c.ca/~sarrazip/dev/conjug_manager.html.

Les données d'apprentissage pour l'anglais, l'espagnol, l'italien, le portugais et le roumain ont été générées en utilisant des techniques d'apprentissage non supervisées en utilisant le modèle français comme modèle à interroger pendant la formation.

— Logiciel libre : licence MIT
— Documentation : https://mlconjug.readthedocs.io.

1.1 Langues prises en charge

— Français
— Anglais
— Espagnol
— Italien
— Portugais
— Roumain

1.2 Fonctionnalités

— API facile à utiliser.
— Inclut des modèles pré-entraînés avec une précision de plus de 99% dans la prédiction de la classe de conjugaison des verbes inconnus.
— Entraînez facilement de nouveaux modèles d'apprentissage automatique ou ajoutez de nouvelles langues.
— Intégrez facilement MLConjug dans vos propres projets.
— Peut être utilisé comme un outil de ligne de commande.

1.3 Crédits

Ce logiciel a été créé avec l'aide de Verbiste et de scikit-learn.

Le logo a été conçu par Zuur.

CHAPITRE 2

Installation

2.1 Version stable

Pour installer MLConjug, exécutez cette commande dans votre terminal :

```
$ pip install mlconjug
```

C'est la méthode préférée pour installer MLConjug, car cette méthode installera toujours la version stable la plus récente.

Si pip n'est pas installé , ce guide d'installation Python peut vous guider tout au long du processus.

2.2 Depuis le code source

Le code source de MLConjug peut être téléchargé à partir du projet GitHub.

Vous pouvez soit cloner le dépôt public :

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

Ou téléchargez une archive :

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

Une fois que vous avez une copie du code source, vous pouvez l'installer avec :

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

CHAPITRE 3

Utilisation

Note : La langue par défaut est le français : lorsqu'il est appelé sans spécifier de langue, le conjugateur va essayer de conjuguer le verbe en français.

Utiliser MLConjug dans un projet avec les modèles de conjugaison pré-entraînés fournis :

```
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)
```

Utiliser MLConjug dans un projet et entraîner un nouveau modèle

```
# 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)
```

Pour utiliser MLConjug depuis la ligne de commande :

```
$ mlconjug manger

$ mlconjug bring -l en

$ mlconjug gallofar --language es
```

CHAPITRE 4

Documentation de l'Api mlconjug

4.1 Référence de l'API pour les classes dans mlconjug.mlconjug.py

Module principal de MLConjug .

Ce module déclare les classes principales avec lesquelles l'utilisateur interagit.

Le module définit les classes nécessaires à l'interface avec les modèles de Machine Learning.

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

Vectorizer optimisé pour extraire les fonctionnalités des verbes.

Le Vectorizer hérite de la classe sklearn.feature_extraction.text.CountVectorizer.

Comme dans les langues indo-européennes les verbes sont infléchis en ajoutant un suffixe morphologique, le vectorizer extrait les terminaisons verbales et d'autres traits caractéristiques et produit une représentation vectorielle du verbe avec des caractéristiques binaires.

Pour améliorer les résultats de l'extraction de caractéristiques, plusieurs autres fonctionnalités ont été incluses :

Les caractéristiques sont la fin du verbe n-grammes, départ n-grammes, la longueur du verbe, le nombre de voyelles, le nombre de consonnes et le rapport des voyelles sur les consonnes.

Paramètres
- **verb** – string. Verbe à vectoriser.
- **lang** – string. Langue à analyser.
- **ngram_range** – tuple. La taille de la fenêtre de n-grams.

Renvoie list. Liste des caractéristiques les plus saillantes du verbe pour la tâche de trouver sa classe de conjugaison.

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

C'est la classe principale du projet.

La classe gère la base de données de Verbiste et fournit une interface avec le modèle scikit-learn.

Si aucun paramètre n'est fourni, la langue par défaut est le français et le modèle de conjugaison pré-entraîné du français est utilisé.

La classe définit la méthode conjugate(verbe, langue) qui est la méthode principale du module.

Paramètres
- **language** – string. Langue du conjugateur. La langue par défaut est "fr" pour le français.
- **model** – mlconjug.Model ou scikit-learn Pipeline ou Classifier implémentant les méthodes fit () et predict (). Un pipeline fourni par l'utilisateur si l'utilisateur a formé son propre pipeline.

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

C'est la méthode principale de cette classe.

La méthode vérifie d'abord si le verbe est dans Verbiste.

Si ce n'est pas le cas, et qu'un modèle d'apprentissage de scikit-learn pré-entraîné a été fourni, la méthode appelle alors le modèle

Renvoie un objet Verb ou None.

Paramètres
- **verb** – string. Verbe à conjuguer.
- **subject** – string. Active les pronoms abrégés ou entiers. La valeur par défaut est "abbrev" Sélectionnez "pronoun" pour les pronoms complets.

Renvoie Objet Verb ou None.

set_model(*model*)

Assigne le modèle scikit-learn pré-entraîné fourni afin de conjuguer les verbes inconnus.

Paramètres **model** – Classificateur ou Pipeline scikit-learn.

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

Cette classe contient et gère l'ensemble de données nécessaires à l'entrainement du modèle d'apprentissage.

Définit les fonctions d'assistance pour la gestion des tâches d'apprentissage automatique, telles que la construction d'un ensemble d'entraînement et de test.

Paramètres **verbs_dict** – Un dictionnaire de verbes et leur classe de conjugaison correspondante.

construct_dict_conjug()

Remplit le dictionnaire contenant les classes de conjugaison.

Remplit les listes contenant les verbes et leurs classes.

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

Divise les données en un ensemble d'entrainement et un ensemble de test.

Paramètres
- **threshold** – int. Taille minimale de la classe de conjugaison à diviser.
- **proportion** – float. Proportion d'échantillons dans l'ensemble d'entraînement. Doit être compris entre 0 et 1.

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

Bases : object

Cette classe gère le modèle scikit-learn.
Le Pipeline comprend un vectoriseur de caractéristiques, un sélecteur de caractéristiques et un classificateur.
la méthode __init__ fournira des valeurs par défaut acceptables qui obtiennent plus de 92% de précision de prédiction en moyenne selon les langues.

Paramètres
- **vectorizer** – scikit-learn Vectorizer.
- **feature_selector** – scikit-learn Classifier avec une méthode fit_transform ()
- **classifier** – scikit-learn Classifier avec une méthode predict ()
- **language** – langue du corpus de verbes à analyser.

train (*samples*, *labels*)
Entraîne le modèle sur les échantillons et les labels fournis.

Paramètres
- **samples** – list. Liste de verbes.
- **labels** – list. Liste de classes de cojugaison de verbes.

predict (*verbs*)
Prédit la classe de conjugaison de la liste de verbes fournie.

Paramètres **verbs** – list. Liste de verbes.

Renvoie list. Liste des classes de conjugaison prédites.

4.2 Référence de l'API pour les classes dans mlconjug.PyVerbiste.py

PyVerbiste.

Une bibliothèque Python pour conjuguer les verbes en français, anglais, espagnol, italien, portugais et roumain (et plus bientôt !).
La librarie contient des données de conjugaison générées par des modèles d'apprentissage automatique utilisant la bibliothèque Python mlconjug.
Plus d'informations sur mlconjug sur https://pypi.org/project/mlconjug/

Les données de conjugaison sont conformes au schéma XML défini par Verbiste.
Plus d'informations sur Verbiste à https://perso.b2b2c.ca/~sarrazip/dev/conjug_manager.html

class mlconjug.PyVerbiste.**ConjugManager** (*language='default'*)
C'est la classe qui manipule et sert d'interface avec les fichiers json.

Paramètres **language** – string. | Langue du conjugateur. La langue par défaut est "fr" pour le français.

_load_verbs (*verbs_file*)
Charge et analyse les verbes du fichier json.

Paramètres **verbs_file** – string ou Path object. Chemin d'accès au fichier json des verbes.

_load_conjugations (*conjugations_file*)
Charge et analyse les conjugaisons à partir du fichier json.

Paramètres **conjugations_file** – string ou objet Path. Chemin d'accès au fichier xml de conjugaison.

_detect_allowed_endings ()

Détecte les terminaisons autorisées des verbes dans les langues prises en charge.

Toutes les langues prises en charge à l'exception de l'anglais limitent la forme qu'un verbe peut prendre.

Comme l'anglais est beaucoup plus productif et varié dans la morphologie de ses verbes, n'importe quel mot est autorisé comme un verbe.

Renvoie set. Un ensemble contenant les terminaisons autorisées des verbes dans la langue cible.

is_valid_verb(*verb*)

Vérifie si le verbe est un verbe valide dans la langue cible.

Les mots anglais sont toujours traités comme des verbes possibles.

Les verbes dans les autres langues sont filtrés par leurs terminaisons.

Paramètres **verb** – string. Le verbe à conjuguer.

Renvoie bool. True si le verbe est un verbe valide dans la langue. False sinon.

get_verb_info(*verb*)

Récupère les informations du verbe et renvoie une instance de VerbInfo.

Paramètres **verb** – string. Verbe à conjuguer.

Renvoie Objet VerbInfo ou None.

get_conjug_info(*template*)

Récupère les informations de conjugaison correspondant à la classe de conjugaison donné.

Paramètres **template** – string. Nom de la classe de conjugaison du verbe.

Renvoie OrderedDict ou None. OrderedDict contenant les suffixes conjugués de la classe de conjugaison.

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

Bases : *mlconjug.PyVerbiste.ConjugManager*

C'est la classe qui manipule et sert d'interface avec les fichiers xml de Verbiste.

Paramètres **language** – string. | Langue du conjugateur. La langue par défaut est "fr" pour le français.

_load_verbs(*verbs_file*)

Charge et analyse les verbes du fichier XML.

Paramètres **verbs_file** – string ou Path object. Chemin d'accès au fichier xml des verbes.

_parse_verbs(*file*)

Analyse le fichier XML.

Paramètres **file** – FileObject. Fichier XML contenant les verbes.

Renvoie OrderedDict. Un OrderedDict contenant le verbe et sa classe de conjugaison pour tous les verbes dans le fichier.

_load_conjugations(*conjugations_file*)

Charge et analyse les conjugaisons à partir du fichier json.

Paramètres **conjugations_file** – string ou objet Path. Chemin d'accès au fichier xml de conjugaison.

_parse_conjugations(*file*)

Analyse le fichier XML.

Paramètres **file** – FileObject. Fichier XML contenant les classes de conjugaison.

Renvoie OrderedDict. Un OrderedDict contenant toutes les classes de conjugaison dans le fichier.

_load_tense(*tense*)

Charge et analyse les formes fléchies du temps à partir du fichier xml.

Paramètres **tense** – liste des balises xml contenant des formes fléchies. La liste des formes fléchies pour le temps courant en cours de traitement.

Renvoie list. Liste de formes conjuguées.

_detect_allowed_endings()

Détecte les terminaisons autorisées des verbes dans les langues prises en charge.

Toutes les langues prises en charge à l'exception de l'anglais limitent la forme qu'un verbe peut prendre.

Comme l'anglais est beaucoup plus productif et varié dans la morphologie de ses verbes, n'importe quel mot est autorisé comme un verbe.

Renvoie set. Un ensemble contenant les terminaisons autorisées des verbes dans la langue cible.

get_conjug_info(*template*)

Récupère les informations de conjugaison correspondant à la classe de conjugaison donné.

Paramètres **template** – string. Nom de la classe de conjugaison du verbe.

Renvoie OrderedDict ou None. OrderedDict contenant les suffixes conjugués de la classe de conjugaison.

get_verb_info(*verb*)

Récupère les informations du verbe et renvoie une instance de VerbInfo.

Paramètres **verb** – string. Verbe à conjuguer.

Renvoie Objet VerbInfo ou None.

is_valid_verb(*verb*)

Vérifie si le verbe est un verbe valide dans la langue cible.

Les mots anglais sont toujours traités comme des verbes possibles.

Les verbes dans les autres langues sont filtrés par leurs terminaisons.

Paramètres **verb** – string. Le verbe à conjuguer.

Renvoie bool. True si le verbe est un verbe valide dans la langue. False sinon.

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

Cette classe définit la structure d'information du verbe Verbiste.

Paramètres

- **infinitive** – string. Forme infinitive du verbe.
- **root** – string. Racine lexicale du verbe.
- **template** – string. Nom de la classe de conjugaison du verbe.

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.

Paramètres

- **verb_info** – Objet VerbInfo.
- **conjug_info** – OrderedDict.
- **subject** – string. Active les pronoms abrégés ou entiers. La valeur par défaut est "abbrev" Sélectionnez "pronoun" pour les pronoms complets.
- **predicted** – bool. Indique si les informations de conjugaison ont été prédites par le modèle ou extraites de l'ensemble de données.

iterate()

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

_load_conjug()

Remplit les formes fléchies du verbe.

Ceci est la version générique de cette méthode.

Cette version n'ajoute pas de pronoms personnels aux formes conjuguées.

Cette méthode peut gérer n'importe quel nouvelle langue si la structure de conjugaison est conforme au schéma XML de Verbiste.

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

Bases : *mlconjug.PyVerbiste.Verb*

Cette classe définit l'objet Verb français.

_load_conjug()

Remplit les formes fléchies du verbe.

Ajoute les pronoms personnels aux verbes fléchis.

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*)

Bases : *mlconjug.PyVerbiste.Verb*

Cette classe définit l'objet de Verb anglais.

_load_conjug()

Remplit les formes fléchies du verbe.

Ajoute les pronoms personnels aux verbes fléchis.

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*)

Bases : *mlconjug.PyVerbiste.Verb*

Cette classe définit l'objet Verb espagnol.

_load_conjug()

Remplit les formes fléchies du verbe.

Ajoute les pronoms personnels aux verbes fléchis.

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*)

Bases : *mlconjug.PyVerbiste.Verb*

Cette classe définit l'objet Verb italien.

_load_conjug()

Remplit les formes fléchies du verbe.

Ajoute les pronoms personnels aux verbes fléchis.

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*)

Bases : *mlconjug.PyVerbiste.Verb*

Cette classe définit l'objet Verb portugais.

_load_conjug()

Remplit les formes fléchies du verbe.

Ajoute les pronoms personnels aux verbes fléchis.

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*)
Bases : *mlconjug.PyVerbiste.Verb*
Cette classe définit l'objet Verb roumain.

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

_load_conjug()

Remplit les formes fléchies du verbe.
Ajoute les pronoms personnels aux verbes fléchis.

CHAPITRE 5

Contribuer

Les contributions sont les bienvenues, et elles sont grandement appréciées ! Chaque petit coup de pouce aide, et le crédit sera toujours donné.

Vous pouvez contribuer de plusieurs façons :

5.1 Types de contributions

5.1.1 Signaler les bugs

Signaler les bugs sur https://github.com/SekouD/mlconjug/issues.

Si vous signalez un bug, veuillez inclure :

- Le nom et la version de votre système d'exploitation.
- Tous les détails sur votre configuration locale qui pourraient être utiles pour le dépannage.
- Les étapes détaillées pour reproduire le bug.

5.1.2 Corriger les bugs

Consultez le depot GitHub pour voir si il y a des bugs en attente de correction. Tout ce qui est étiqueté « bug » et « help wanted » est ouvert à quiconque veut l'implémenter.

5.1.3 Implémenter de nouvelles fonctionnalités

Consultez GitHub pour voir si le projet est en attente de nouvelles fonctionnalités. Tout ce qui est étiqueté avec « enhancement » et « help wanted » est ouvert à quiconque veut l'implémenter.

5.1.4 Ecrire de la documentation

MLConjug pourrait toujours utiliser plus de documentation, que ce soit dans la documentation officielle de MLConjug, dans les docstrings, ou même avec une mention sur des blogs, articles et autres.

5.1.5 Soumettre des commentaires

La meilleure façon d'envoyer vos commentaires est de remplir une « issue » sur https://github.com/SekouD/mlconjug/issues.

Si vous proposez une fonctionnalité :

— Expliquez en détail comment cela fonctionnerait.
— Gardez la portée aussi minime que possible, pour rendre la fonctionnalité plus facile à mettre en œuvre.
— Rappelez-vous qu'il s'agit d'un projet mené par des bénévoles, et que les contributions sont les bienvenues :)

5.2 Commencez !

Prêt à contribuer ? Voici comment configurer *mlconjug* pour le développement local.

1. Forkez le dépôt *mlconjug* sur GitHub.
2. Cloner votre fork localement :

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

3. Installez votre copie locale dans un virtualenv. Si vous avez installé virtualenvwrapper, voici comment configurer votre fork pour le développement :

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

4. Créer une branche pour le développement local :

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

   Vous pouvez maintenant faire vos changements localement.

5. Lorsque vous avez fini de faire vos changements, vérifiez que vos changements passent flake8 et les tests, y compris les test d'autres versions de Python avec tox :

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

   Pour obtenir flake8 et tox, il suffit de les installer dans votre virtualenv.

6. Validez vos modifications en faisant un commit et faites un push de votre branche sur GitHub :

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

7. Soumettre une requête pull (pull request) sur le site Web de GitHub.

5.3 Directives sur les Pull Request

Avant de soumettre une requête pull, vérifiez qu'elle respecte les consignes suivantes :

1. La requête pull doit inclure des tests.
2. Si la requête pull ajoute des fonctionnalités, les documents doivent être mis à jour. Mettez votre nouvelle fonctionnalité dans une fonction avec une docstring et ajoutez la fonctionnalité à la liste dans README.rst.
3. La requête pull devrait fonctionner pour Python 3.3, 3.4, 3.5 et 3.6. Vérifiez sur https://travis-ci.org/SekouD/mlconjug/pull_requests et assurez-vous que les tests passent pour toutes les versions de Python supportées.

5.4 Conseils

Pour exécuter un sous-ensemble de tests :

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

CHAPITRE 6

Crédits

6.1 Responsable du développement

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

6.2 Contributeurs

— Le logo a été conçu par Zuur.

CHAPITRE 7

Historique

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)

— Mise à jour Type annotations pour conformité PEP-561.

7.10 3.1.1 (2018-06-26)

— Amélioration mineure de l'Api (voir documentation de l'API)

7.11 3.1.0 (2018-06-24)

— Mise à jour des modèles de conjugaison pour l'espagnol et le portugais.
— Changements internes du format des données verbistes de xml à json pour une meilleure gestion des caractères Unicode.
— Nouvelle classe ConjugManager pour ajouter plus facilement de nouvelles langues à mlconjug.
— Amélioration mineure de l'Api (voir documentation de l'API)

7.12 3.0.1 (2018-06-22)

— **Mise à jour de tous les modèles de prédiction pré-entrainés fournis :**
  — Implémentation d'un nouveau vectrorizer extrayant des fonctionnalités plus significatives.
  — En conséquence, les performances des modèles ont nettement augmenté dans toutes les langues.
  — Le rappel et la précision sont intimement proches de 100%. L'anglais étant le seul à atteindre un score parfait à la fois en Recall et Precision.
— **Principales modifications de l'API :**
  — J'ai supprimé la classe EndingCustomVectorizer et refactorisé sa fonctionnalité dans une fonction de niveau supérieur appelée extract_verb_features ()
  — Le nouveau modèle amélioré fourni est maintenant compressé par zip avant la publication, car la taille des modèles a tellement augmenté que leur taille les rendait peu pratiques à distribuer avec le paquet.
  — Renommé « Model.model » en « Model.pipeline »

- Renommé « DataSet.liste_verbes » et « DataSet.liste_templates » à « DataSet.verbs_list » et « DataSet.templates_list » respectivement. (Excusez mon français ;-))
- Ajout des attributs « predicted » et « confidence_score » à la classe Verb.
- L'ensemble du package a été annoté avec les informations sur le type de tous les objets.

7.13 2.1.11 (2018-06-21)

- **Mise à jour de tous les modèles de prédiction pré-entrainés fournis**
  - Le Conjugueur français a une précision d'environ 99,94% dans la prédiction de la classe de conjugaison correcte d'un verbe français. C'est la base de référence car je travaille dessus depuis un certain temps maintenant.
  - Le Conjugueur anglais a une précision d'environ 99,78% dans la prédiction de la classe de conjugaison correcte d'un verbe anglais. C'est l'une des plus grandes améliorations depuis la version 2.0.0
  - Le Conjugueur espagnol a une précision d'environ 99,65% dans la prédiction de la classe de conjugaison correcte d'un verbe espagnol. Il a également vu une amélioration considérable depuis la version 2.0.0
  - Le Conjugateur roumain a une précision d'environ 99,06% pour prédire la bonne classe de conjugaison d'un verbe roumain. C'est de loin le gain le plus important. J'ai modifié le vectorizer pour mieux prendre en compte les caractéristiques morphologiques ou les verbes roumains. (Le score précédent était d'environ 86%, donc ce sera sympa pour nos amis roumains d'avoir un conjugateur de confiance)
  - Le Conjugueur portugais a une précision d'environ 96,73% pour prédire la bonne classe de conjugaison d'un verbe portugais.
  - Le Conjugueur Italien a une précision d'environ 94.05% dans la prédiction de la classe de conjugaison correcte d'un verbe italien.

7.14 2.1.9 (2018-06-21)

- **Maintenant, le Conjugateur ajoute des informations supplémentaires à l'objet Verbe renvoyé.**
  - Si le verbe considéré est déjà dans Verbiste, la conjugaison pour le verbe est récupérée directement de la mémoire.
  - Si le verbe à l'étude est inconnu dans Verbiste, la classe Conjugator définit maintenant l'attribut booléen "predicted" et l'attribut float "confidence_score" à l'instance de l'objet Verb renvoyé par Conjugator.conjugate (verbe).
- Ajout de Type annotations à la bibliothèque entière pour garantir la robustesse du programme.
- La performance des modèles anglais et roumain s'est considérablement améliorée ces derniers temps. Je suppose que dans quelques itérations, ils seront à égalité avec le modèle français qui est le plus performant en ce moment car j'ai réglé ses paramètres depuis pas mal de temps maintenant.
- Amélioration de la localisation du programme.
- Maintenant, l'interface utilisateur de mlconjug est disponible en français, espagnol, italien, portugais et roumain, en plus de l'anglais.
- Toute la documentation du projet a été traduite dans les langues supportées.

7.15 2.1.5 (2018-06-15)

- Localisation ajoutée.
- Maintenant, l'interface utilisateur de mlconjug est disponible en français, espagnol, italien, portugais et roumain, en plus de l'anglais.

7.16 2.1.2 (2018-06-15)

— Ajout de la détection des verbes invalides.

7.17 2.1.0 (2018-06-15)

— Mise à jour de tous les modèles d'apprentissage de langue pour compatibilité avec scikit-learn 0.19.1.

7.18 2.0.0 (2018-06-14)

— Inclut le modèle de conjugaison anglais.
— Inclut le modèle de conjugaison espagnol.
— Inclut le modèle de conjugaison italien.
— Inclut le modèle de conjugaison portugais.
— Inclut le modèle de conjugaison roumain.

7.19 1.2.0 (2018-06-12)

— Refactorisé l'API, une classe unique Conjugateur est maintenant suffisante pour l'interface avec le module.
— Inclut un modèle de conjugaison du français amélioré.
— Ajout du support pour plusieurs langues.

7.20 1.1.0 (2018-06-11)

— Refactorisé l'API, une classe unique Conjugateur est maintenant suffisante pour l'interface avec le module.
— Inclut un modèle de conjugaison du français amélioré.

7.21 1.0.0 (2018-06-10)

— Première version sur PyPI.

CHAPITRE 8

Indices et tables

— genindex
— modindex
— search

Index des modules Python

m

Index

Symboles

C

D

E

G

I

M

P

S

T

V