MLConjug Documentation

Versión 3.4.0

SekouD

29 de abril de 2019

Índice general

Contenido:

CAPÍTULO 1

mlconjug

Una biblioteca de Python para conjugar verbos en francés, inglés, español, italiano, portugués y rumano (más pronto) usando técnicas de Machine Learning.
Cualquier verbo en uno de los idiomas soportados puede ser conjugado, ya que el módulo contiene un modelo de Aprendizaje automático sobre cómo se comportan los verbos.
Incluso los verbos completamente nuevos o inventados pueden conjugarse exitosamente de esta manera.
Los modelos preentrenados suministrados se componen de:

- un extractor de funciones binarias,
- un selector de funciones usando Linear Support Vector Classification,
- un clasificador que usa el Gradiente de Gradiente Estocástico.

MLConjug usa scikit-learn para implementar los algoritmos de Machine Learning.
Los usuarios de la biblioteca pueden usar cualquier clasificador compatible de scikit-learn para modificar y volver a entrenar el modelo.

Los datos de capacitación se basan en Verbiste https://perso.b2b2c.ca/~sarrazip/dev/conjug_manager.html.

Los datos de entrenamiento para inglés, español, italiano, portugués y rumano se generaron utilizando técnicas de aprendizaje no supervisadas utilizando el modelo francés como modelo para consultar durante el entrenamiento.

- Software libre: licencia MIT
- Documentación: https://mlconjug.readthedocs.io

1.1 Idiomas admitidos

- Francés
- Inglés
- Español
- Italiano
- Portugués
- Rumano

1.2 Caracteristicas

- API fácil de usar.
- Incluye modelos pre-entrenados con un 99 % de precisión en la predicción de la conjugación de verbos desconocidos.
- Entrene fácilmente nuevos modelos o agregue nuevos idiomas.
- Integre fácilmente MLConjug en sus propios proyectos.
- Se puede usar como una herramienta de línea de commandos.

1.3 Créditos

Este paquete fue creado con la ayuda de Verbiste y scikit-learn.

El logo fue diseñado por Zuur.

CAPÍTULO 2

Instalación

2.1 Lanzamiento estable

Para instalar MLConjug, ejecute este comando en su terminal:

```
$ pip install mlconjug
```

Este es el método preferido para instalar MLConjug, ya que siempre instalará la versión estable más reciente»

Si no tiene‘ pip‘_ instalado, esta guía de instalación de Python puede guiarlo a través del proceso.

2.2 De fuentes

Las fuentes para MLConjug se pueden descargar del‘ Github repo‘_ «

Puedes clonar el repositorio público:

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

O descargue el‘ tarball‘_:

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

Una vez que tenga una copia de la fuente, puede instalarla con:

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

CAPÍTULO 3

Uso

Nota: El idioma predeterminado es el francés. Cuando se invoca sin especificar un idioma, la biblioteca intentará conjugar el verbo en francés»

Utilizar MLConjug en un proyecto con los modelos de conjugación pre-entrenados proporcionados

```
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)
```

Usar MLConjug en un proyecto y entrenar un nuevo modelo

```
# 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)
```

Para usar MLConjug desde la línea de comando

```
$ mlconjug manger

$ mlconjug bring -l en

$ mlconjug gallofar --language es
```

CAPÍTULO 4

Package Api Documentation for mlconjug

4.1 Referencia de API para las clases en mlconjug.mlconjug.py

Módulo principal MLConjug

Este módulo declara las principales clases con las que el usuario interactúa»

El módulo define las clases necesarias para interactuar con los modelos de Machine Learning»

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

Vectorizador personalizado optimizado para extraer las características de los verbos»
Las subclases Vectorizer sklearn.feature_extraction.text.CountVectorizer»
el vectorizador extrae las terminaciones de los verbos y produce una representación vectorial del verbo con características binarias

Para mejorar los resultados de la extracción de características, se han incluido varias otras características:

Las características son los n-gramas finales del verbo, los n-grams iniciales, la longitud del verbo, el número de vocales, el número de consonantes y la proporción de vocales sobre consonantes.

Parámetros

- **verb** – string. Verbo para vectorizar
- **lang** – string. Idioma para analizar.

- **ngram_range** – tupla El rango de la ventana deslizante ngram.

Devuelve list. Lista de las características más destacadas del verbo para la tarea de encontrar su clase de conjugación.

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

Esta es la clase principal del Proyecto.
La clase maneja el conjunto de datos Verbiste y proporciona una interfaz con el modelo scikit-learn.
Si no se proporcionan parámetros, el idioma predeterminado es francés y se utiliza el modelo de conjugación francés preentrenado.
La clase define el método conjugado (verbo, lenguaje) que es el método principal del modulo.

Parámetros

- **language** – string. Idioma del conjugador. El idioma predeterminado es "fr" para francés.
- **model** – mlconjug.Model o scikit-learn Pipeline o Classifier implementando los métodos fit () y predict (). Un conducto proporcionado por el usuario si el usuario ha entrenado su propia tubería.

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

Este es el método principal de esta clase.
Primero verifica si el verbo está en Verbiste.
Si no es así, y se ha proporcionado un modelo de aprendizaje de scikit preentrenado, el método llama al modelo.

Devuelve un objeto Verb o Ninguno.

Parámetros

- **verb** – string. Verbo para conjugar.
- **subject** – string. Alterna los pronombres abreviados o completos. El valor predeterminado es "abreviación". Seleccione "pronombre" para los pronombres completos.

Devuelve Verb object or None.

set_model(*model*)
Asigna el modelo de aprendizaje de scikit preentrenado provisto para poder conjugar verbos desconocidos.

Parámetros **model** – scikit-learn Classifier o Pipeline»

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

Esta clase contiene y administra el conjunto de datos»
Define funciones auxiliares para administrar tareas de Aprendizaje automático, como la construcción de un conjunto de entrenamiento y prueba.

Parámetros **verbs_dict** – Un diccionario de verbos y su correspondiente clase de conjugación.

construct_dict_conjug()

Rellena el diccionario que contiene las plantillas de conjugación.
Rellena las listas que contienen los verbos y sus plantillas.

split_data (*threshold=8*, *proportion=0.5*)
Divide los datos en un conjunto de entrenamiento y prueba.

Parámetros

- **threshold** – int. Tamaño mínimo de la clase de conjugación que se dividirá.
- **proportion** – float. Proporción de muestras en el conjunto de entrenamiento. Debe estar entre 0 y 1.

class mlconjug.mlconjug.**Model** (*vectorizer=None*, *feature_selector=None*, *classifier=None*, *language=None*)
Bases:: clase:‘ objeto‘

Esta clase maneja el modelo scikit-learn.
The Pipeline incluye un vectorizador de funciones, un selector de funciones y un clasificador»
El método __init__ proporcionará buenos valores predeterminados que obtienen más del 92 % de precisión de predicción.

Parámetros

- **vectorizer** – scikit-learn Vectorizer»
- **feature_selector** – clasificador scikit-learn con un método fit_transform ()
- **classifier** – clasificador scikit-learn con un método de predicción ()
- **language** – lenguaje del corpus de verbos a analizar.

train (*samples*, *labels*)
Entrena el modelo en las muestras y etiquetas suministradas.

Parámetros

- **samples** – lista. Lista de verbos.
- **labels** – lista. Lista de plantillas de verbos.

predict (*verbs*)
Predice la clase de conjugación de la lista de verbos proporcionada.

Parámetros **verbs** – lista. Lista de verbos.

Devuelve lista. Lista de grupos de conjugación predichos.

4.2 Referencia de API para las clases en mlconjug.PyVerbiste.py

PyVerbiste.

Una biblioteca de Python para conjugar verbos en francés, inglés, español, italiano, portugués y rumano (más pronto).
Contiene datos de conjugación generados por modelos de aprendizaje automático utilizando la biblioteca de python mlconjug.
Más información sobre mlconjug en https://pypi.org/project/mlconjug/

Los datos de conjugación se ajustan al esquema XML definido por Verbiste.
Más información sobre Verbiste en https://perso.b2b2c.ca/~sarrazip/dev/conjug_manager.html

class mlconjug.PyVerbiste.**ConjugManager**(*language='default'*)
Esta es la clase que maneja los archivos json.

Parámetros **language** – string. | Idioma del conjugador. El idioma predeterminado es "fr" para francés.

_load_verbs(*verbs_file*)
Cargar y analizar los verbos del archivo json.

Parámetros **verbs_file** – string o objeto de ruta. Ruta al archivo json de verbos.

_load_conjugations(*conjugations_file*)
Cargar y analizar las conjugaciones del archivo json.

Parámetros **conjugations_file** – string o objeto de ruta. Ruta al archivo xml de conjugación

_detect_allowed_endings()

Detecta las terminaciones permitidas para los verbos en los idiomas admitidos
Todos los idiomas admitidos, excepto el inglés, restringen la forma que puede adoptar un verbo
Como el inglés es mucho más productivo y variado en la morfología de sus verbos, cualquier palabra se permite como verbo

Devuelve set. Un conjunto que contiene las terminaciones permitidas de los verbos en el idioma de destino

is_valid_verb(*verb*)

Comprueba si el verbo es un verbo válido en el idioma dado.
Las palabras en inglés siempre se tratan como posibles verbos.
Los verbos en otros idiomas se filtran por sus terminaciones.

Parámetros **verb** – string. El verbo conjugar

Devuelve bool. Verdadero si el verbo es un verbo válido en el idioma. De lo contrario, falso»

get_verb_info(*verb*)
Obtiene información del verbo y devuelve una instancia de VerbInfo»

Parámetros **verb** – string. Verbo para conjugar.

Devuelve Objeto VerbInfo o Ninguno»

get_conjug_info(*template*)
Obtiene la información de conjugación correspondiente a la plantilla dada»

Parámetros **template** – string. Nombre del patrón de terminación del verbo»

Devuelve OrderedDict or None. OrderedDict que contiene los sufijos conjugados de la plantilla»

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

Esta es la clase que maneja los archivos Verbiste xml»

Parámetros **language** – string. | Idioma del conjugador. El idioma predeterminado es "fr" para francés.

_load_verbs(*verbs_file*)
Cargar y analizar los verbos del archivo xml»

Parámetros **verbs_file** – cadena o objeto de ruta. Ruta al archivo xml de verbos»

_parse_verbs(*file*)
Analiza el archivo XML»

Parámetros **file** – FileObject. Archivo XML que contiene los verbos»

Devuelve OrderedDict. Un OrderedDict que contiene el verbo y su plantilla para todos los verbos en el archivo»

_load_conjugations(*conjugations_file*)
Cargar y analizar las conjugaciones del archivo json.

Parámetros **conjugations_file** – string o objeto de ruta. Ruta al archivo xml de conjugación

_parse_conjugations(*file*)
Analiza el archivo XML»

Parámetros **file** – FileObject. Archivo XML que contiene las plantillas de conjugación»

Devuelve OrderedDict. Un OrderedDict que contiene todas las plantillas de conjugación en el archivo»

_load_tense(*tense*)
Cargar y analizar las formas del tiempo inflexión del archivo xml»

Parámetros **tense** – lista de etiquetas xml que contienen formas flexionadas La lista de formas declinadas para el tiempo actual que se está procesando.

Devuelve lista. Lista de formas declinadas.

_detect_allowed_endings()

Detecta las terminaciones permitidas para los verbos en los idiomas admitidos
Todos los idiomas admitidos, excepto el inglés, restringen la forma que puede adoptar un verbo
Como el inglés es mucho más productivo y variado en la morfología de sus verbos, cualquier palabra se permite como verbo

Devuelve set. Un conjunto que contiene las terminaciones permitidas de los verbos en el idioma de destino

get_conjug_info(*template*)
Obtiene la información de conjugación correspondiente a la plantilla dada»

Parámetros **template** – string. Nombre del patrón de terminación del verbo»

Devuelve OrderedDict or None. OrderedDict que contiene los sufijos conjugados de la plantilla»

get_verb_info(*verb*)
Obtiene información del verbo y devuelve una instancia de VerbInfo»

Parámetros **verb** – string. Verbo para conjugar.

Devuelve Objeto VerbInfo o Ninguno»

is_valid_verb(*verb*)

Comprueba si el verbo es un verbo válido en el idioma dado.
Las palabras en inglés siempre se tratan como posibles verbos.
Los verbos en otros idiomas se filtran por sus terminaciones.

Parámetros **verb** – string. El verbo conjugar

Devuelve bool. Verdadero si el verbo es un verbo válido en el idioma. De lo contrario, falso»

class mlconjug.PyVerbiste.**VerbInfo**(*infinitive*, *root*, *template*)
Esta clase define la estructura de información del verbo Verbiste»

Parámetros

- **infinitive** – cadena. Forma infinitiva del verbo»
- **root** – cadena. Raíz léxica del verbo»
- **template** – string. Nombre del patrón de terminación 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.

Parámetros

- **verb_info** – Objeto VerbInfo»
- **conjug_info** – OrderedDict»
- **subject** – string. Alterna los pronombres abreviados o completos. El valor predeterminado es "abreviación". Seleccione "pronombre" para los pronombres completos.
- **predicted** – bool. Indica si la información de conjugación fue predicha por el modelo o recuperada del conjunto de datos.

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

_load_conjug()

Rellena las formas flexionadas del verbo»
Esta es la versión genérica de este método»
No agrega pronombres personales a las formas conjugadas»
Este método puede manejar cualquier idioma nuevo si la estructura de conjugación se ajusta al esquema XML de Verbiste»

class mlconjug.PyVerbiste.**VerbFr**(*verb_info*, *conjug_info*, *subject='abbrev'*, *predicted=False*)
Bases:: clase:‘ mlconjug.PyVerbiste.Verb‘

Esta clase define el objeto del verbo francés»

_load_conjug()

Rellena las formas flexionadas del verbo»
Agrega pronombres personales a los verbos inflexionados»

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:: clase:‘ mlconjug.PyVerbiste.Verb‘

Esta clase define el objeto del verbo inglés»

_load_conjug()

Rellena las formas flexionadas del verbo»
Agrega pronombres personales a los verbos inflexionados»

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:: clase:‘ mlconjug.PyVerbiste.Verb‘

Esta clase define el objeto del verbo español»

_load_conjug()

Rellena las formas flexionadas del verbo»
Agrega pronombres personales a los verbos inflexionados»

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:: clase:‘ mlconjug.PyVerbiste.Verb‘

Esta clase define el objeto del verbo italiano»

_load_conjug()

Rellena las formas flexionadas del verbo»
Agrega pronombres personales a los verbos inflexionados»

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:: clase:‘ mlconjug.PyVerbiste.Verb‘

Esta clase define el objeto del verbo portugués»

_load_conjug()

Rellena las formas flexionadas del verbo»
Agrega pronombres personales a los verbos inflexionados»

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:: clase:‘ mlconjug.PyVerbiste.Verb‘

Esta clase define el objeto del verbo rumano»

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

_load_conjug()

Rellena las formas flexionadas del verbo»
Agrega pronombres personales a los verbos inflexionados»

CAPÍTULO 5

Contribuir

Las contribuciones son bienvenidas, y son muy apreciadas. Todo ayuda, y el crédito siempre se dará»

Puedes contribuir de muchas maneras:

5.1 Tipos de contribuciones

5.1.1 Reportar errores

Informar de errores en https://github.com/SekouD/mlconjug/issues»

Si informa un error, por favor incluya:

- El nombre y la versión de su sistema operativo»
- Cualquier detalle sobre su configuración local que pueda ser útil para solucionar problemas»
- Pasos detallados para reproducir el error»

5.1.2 Corregir errores

Revise los problemas de GitHub para detectar errores. Cualquier cosa etiquetada con » error «y » ayuda deseada «está abierta para quien quiera implementarla.

5.1.3 Implementar características

Revise los problemas de GitHub para ver las características. Todo lo que esté etiquetado con » mejora «y » ayuda deseada «está abierto a quien quiera implementarlo.

5.1.4 Escribir documentación

MLConjug siempre podría usar más documentación, ya sea como parte de los documentos oficiales de MLConjug, en docstrings, o incluso en la web en publicaciones de blogs, artículos, etc.

5.1.5 Enviar comentarios

La mejor forma de enviar comentarios es presentar un problema en https://github.com/SekouD/mlconjug/issues»

Si estás proponiendo una función:

- Explica en detalle cómo funcionaría»
- Mantenga el alcance lo más estrecho posible, para que sea más fácil de implementar»
- Recuerda que este es un proyecto impulsado por voluntarios, y que las contribuciones son bienvenidas :)

5.2 ¡Empezar!

¿Listo para contribuir? Así es cómo configurar‘ mlconjug‘ para el desarrollo local «

1. Tenedor el repositorio‘ mlconjug‘ en GitHub.
2. Clona tu tenedor localmente:

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

3. Instale su copia local en un virtualenv. Suponiendo que tiene virtualenvwrapper instalado, así es como configura su tenedor para el desarrollo local

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

4. Crear una sucursal para el desarrollo local

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

   Ahora puedes hacer tus cambios localmente»

5. Cuando hayas terminado de hacer cambios, verifica que tus cambios pasen a flake8 y las pruebas, incluida la prueba de otras versiones de Python con tox

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

   Para obtener flake8 y tox, solo pip instálalos en tu virtualenv»

6. Confirme sus cambios y envíe su sucursal a GitHub

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

7. Presentar una solicitud de extracción a través del sitio web de GitHub»

5.3 Pautas de solicitud de extracción

Antes de enviar una solicitud de extracción, verifique que cumpla con estas pautas:

1. La solicitud de extracción debe incluir pruebas»
2. Si la solicitud de extracción agrega funcionalidad, los documentos deben actualizarse. Ponga su nueva funcionalidad en una función con un docstring, y agregue la función a la lista en README.rst.
3. La solicitud de extracción debería funcionar para Python 3.3, 3.4, 3.5 y 3.6. Consulte https://travis-ci.org/SekouD/mlconjug/pull_requests y asegúrese de que las pruebas se aprueben para todas las versiones compatibles de Python.

5.4 Consejos

Para ejecutar un subconjunto de pruebas

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

CAPÍTULO 6

Créditos

6.1 Líder de desarrollo

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

6.2 Colaboradores

- El logo fue diseñado por Zuur.

CAPÍTULO 7

Historia

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)

- Se actualizaron Type annotations a toda la biblioteca para cumplir con PEP-561.

7.10 3.1.1 (2018-06-26)

- Mejora de API menor (ver API documentation)

7.11 3.1.0 (2018-06-24)

- Actualizado los modelos de conjugación para español y portugués.
- Cambios internos en el formato de los datos verbales de xml a json para un mejor manejo de los caracteres Unicode.
- Nueva clase ConjugManager para agregar más fácilmente nuevos lenguajes a mlconjug.
- Mejora de API menor (ver API documentation)

7.12 3.0.1 (2018-06-22)

- **Se actualizaron todos los modelos de predicción preparados previamente:**
    - Implementado un nuevo vectrorizer que extrae características más significativas.
    - Como resultado, el rendimiento de los modelos ha aumentado en todos los idiomas.
    - La recuperación y la precisión están íntimamente cerca del 100 %. El inglés es el único en lograr un puntaje perfecto en Recall y Precision.
- **Principales cambios de API:**
    - Eliminé la clase EndingCustomVectorizer y refactoré su funcionalidad en una función de nivel superior llamada extract_verb_features ()
    - El nuevo modelo mejorado que se proporciona ahora se comprime en zip antes de su lanzamiento porque el espacio de características ha crecido tanto que su tamaño los hizo poco prácticos para distribuir con el paquete.
    - Renombrado «Model.model» a «Model.pipeline»
    - Renombrado «DataSet.liste_verbes» y «DataSet.liste_templates» a «DataSet.verbs_list» y «DataSet.templates_list» respectivamente. (Perdón, mi francés ;-))
    - Se agregaron los atributos «predicha» y «confidence_score» al verbo de clase.
    - El paquete completo ha sido escrito a máquina. Pronto agregaré los stubs de tipo de mlconjug a la clase de mecanografía.

7.13 2.1.11 (2018-06-21)

- **Se actualizaron todos los modelos de predicción pre-entrenados provistos**
    - El Conjugador francés tiene una precisión de aproximadamente 99.94 % en la predicción de la clase de conjugación correcta de un verbo francés. Esta es la línea de base ya que he estado trabajando en ella desde hace un tiempo.
    - El Conjugador inglés tiene una precisión de aproximadamente 99.78 % en la predicción de la clase correcta de conjugación de un verbo inglés. Esta es una de las mejoras más grandes desde la versión 2.0.0
    - El Conjugador español tiene una precisión de aproximadamente 99.65 % en la predicción de la clase de conjugación correcta de un verbo español. También ha visto una mejora considerable desde la versión 2.0.0
    - El Conjugador rumano tiene una precisión de aproximadamente 99.06 % en la predicción de la clase de conjugación correcta de un verbo rumano. Esta es, de lejos, la mayor ganancia. Modifiqué el vectorizador para tener mejor en cuenta las características morfológicas o los verbos rumanos. (el puntaje anterior fue de alrededor del 86 %, por lo que será bueno que nuestros amigos rumanos tengan un conjugador de confianza)
    - El Conjugador portugués tiene una precisión de aproximadamente 96.73 % en la predicción de la clase de conjugación correcta de un verbo portugués.
    - El Conjugador italiano tiene una precisión de aproximadamente 94.05 % en la predicción de la clase de conjugación correcta de un verbo italiano.

7.14 2.1.9 (2018-06-21)

- **Ahora el Conjugador agrega información adicional al objeto Verb devuelto.**
    - Si el verbo en cuestión ya está en Verbiste, la conjugación para el verbo se recupera directamente de la memoria.
    - Si el verbo en consideración es desconocido en Verbiste, la clase Conjugador ahora establece el atributo booleano "predicha" y el puntaje de confianza del atributo flotante en la instancia del objeto Verb que devuelve Conjugator.conjugate (verbo).
- Se agregó Type annotations a toda la biblioteca para mayor robustez y facilidad de escalamiento.
- El rendimiento de los modelos en inglés y rumano ha mejorado significativamente últimamente. Supongo que en algunas iteraciones más estarán a la par con el modelo francés, que es el mejor en este momento, ya que he estado ajustando sus parámetros por un año. No tanto con los otros idiomas, pero si actualiza regularmente verá agradables improvisaciones en la versión 2.2.
- Mejorado la localización del programa.
- Ahora la interfaz de usuario de mlconjug está disponible en francés, español, italiano, portugués y rumano, además del inglés»
- Toda la documentación del proyecto ha sido traducida a los idiomas admitidos.

7.15 2.1.5 (2018-06-15)

- Localización agregada»
- Ahora la interfaz de usuario de mlconjug está disponible en francés, español, italiano, portugués y rumano, además del inglés»

7.16 2.1.2 (2018-06-15)

- Se agregó detección de verbos inválidos»

7.17 2.1.0 (2018-06-15)

- Actualicé todos los modelos de idiomas para compatibilidad con scikit-learn 0.19.1»

7.18 2.0.0 (2018-06-14)

- Incluye el modelo de conjugación en inglés»
- Incluye el modelo de conjugación español»
- Incluye el modelo de conjugación italiano»
- Incluye el modelo de conjugación portugués»
- Incluye el modelo de conjugación rumano»

7.19 1.2.0 (2018-06-12)

- Refactorizó la API. Ahora se necesita un Conjugador de una sola clase para interactuar con el módulo»
- Incluye un modelo mejorado de conjugación en francés»
- Se agregó soporte para múltiples idiomas»

7.20 1.1.0 (2018-06-11)

- Refactorizó la API. Ahora se necesita un Conjugador de una sola clase para interactuar con el módulo»
- Incluye un modelo mejorado de conjugación en francés»

7.21 1.0.0 (2018-06-10)

- Primer lanzamiento en PyPI»

CAPÍTULO 8

Índices y tablas

- genindex
- modindex
- search

Índice de Módulos Python

m

Índice

Símbolos

C

D

E

G

I

M