MLConjug Documentation

Versão 3.4.0

SekouD

29 abr, 2019

Conteúdo

Conteúdo:

CAPÍTULO 1

mlconjug

Uma biblioteca Python para conjugar verbos em francês, inglês, espanhol, italiano, português e romeno (mais em breve) usando técnicas de Machine Learning.
Qualquer verbo em uma das linguagens suportadas pode ser conjugado, já que o módulo contém um modelo de Aprendizado de Máquina de como os verbos se comportam.
Mesmo verbos completamente novos ou inventados podem ser conjugados com sucesso desta maneira.
Os modelos pré-treinados fornecidos são compostos por:

- um extrator de recurso binário
- um seletor de recursos usando a Classificação Linear de Vetor de Suporte
- um classificador usando o descendente de gradiente estocástico»

O MLConjug usa o scikit-learn para implementar os algoritmos de 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 .
Os dados de treinamento para inglês, espanhol, italiano, português e romeno foram gerados usando técnicas de aprendizado não supervisionadas usando o modelo francês como modelo para consulta durante o treinamento.

- Software livre: licença do MIT
- Documentação: https://mlconjug.readthedocs.io.

1.1 Idiomas Suportados

- Francês
- Inglês
- Espanhol
- Italiano
- Português
- Romena

1.2 Características

- Fácil de usar API»
- Inclui modelos pré-treinados com precisão de 99% + na previsão de classe de conjugação de verbos desconhecidos.
- Treine facilmente novos modelos ou adicione novos idiomas.
- Integre facilmente o MLConjug em seus próprios projetos.
- Pode ser usado como uma ferramenta de linha de comando.

1.3 Créditos

Este pacote foi criado com a ajuda de Verbiste e scikit-learn.

O logotipo foi projetado por Zuur.

CAPÍTULO 2

Instalação

2.1 Versão estável

Para instalar o MLConjug, execute este comando no seu terminal:

```
$ pip install mlconjug
```

Este é o método preferido para instalar o MLConjug, já que ele sempre instalará a versão estável mais recente.

Se você não tem o‘ pip‘_ instalado, este guia de instalação do Python pode guiá-lo através do processo.

2.2 De fontes

As fontes do MLConjug podem ser baixadas do repositório do Github.

Você pode clonar o repositório público:

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

Ou baixe o‘ tarball‘_:

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

Depois de ter uma cópia da fonte, você pode instalá-lo com:

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

CAPÍTULO 3

Uso

Nota: O idioma padrão é o francês. Quando chamado sem especificar um idioma, a biblioteca tentará conjugar o verbo em francês.

Usar o MLConjug em um projeto com os modelos de conjugação pré-treinados fornecidos

```
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)
```

Para usar o MLConjug em um projeto e treinar um novo 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 o MLConjug na linha de comando

```
$ mlconjug manger

$ mlconjug bring -l en

$ mlconjug gallofar --language es
```

CAPÍTULO 4

Package Api Documentation for mlconjug

4.1 Referência de API para as classes em mlconjug.mlconjug.py

Módulo principal do MLConjug.

Este módulo declara as principais classes com as quais o usuário interage.

O módulo define as classes necessárias para interagir com os modelos de Machine Learning.

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

Custom Vectorizer otimizado para extrair recursos de verbos.
O vetorizador subclasse 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.

Para aprimorar os resultados da extração de recursos, vários outros recursos foram incluídos:

As características são o final do verbo n-grams, iniciando n-grams, comprimento do verbo, número de vogais, número de consoantes e a razão entre vogais e consoantes.

Parâmetros

- **verb** – string. Verb para vetorizar.
- **lang** – string. Language to analyze.

- **ngram_range** – tupla. O intervalo da janela deslizante do ngram.

Retorno Lista. Lista das características mais salientes do verbo para a tarefa de encontrar sua classe de conjugação.

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

Esta é a principal classe do projeto.

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.

A classe define o método conjugado (verbo, idioma) que é o método principal do módulo.

Parâmetros

- **language** – string. Linguagem do conjugador. O idioma padrão é "fr" para francês.
- **model** – mlconjug.Model ou scikit-learn Pipeline ou Classifier que implementam os métodos fit () e predict (). Um usuário forneceu o pipeline se o usuário tiver treinado seu próprio pipeline.

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

Este é o principal método desta classe.

Primeiro, verifica se o verbo está em 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.

Retorna um objeto Verbo ou Nenhum.

Parâmetros

- **verb** – string. Verb para conjugar.
- **subject** – string. Alterna os pronomes abreviados ou completos. O valor padrão é "abrev". Selecione "pronome" para os pronomes completos.

Retorno Objeto Verbo ou Nenhum.

set_model(*model*)

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

Parâmetros **model** – Classificador ou Pipeline do scikit-learn.

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

Esta classe mantém e gerencia o conjunto de dados.

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

Parâmetros **verbs_dict** – Um dicionário de verbos e sua classe de conjugação correspondente.

construct_dict_conjug()

Popula o dicionário que contém os modelos de conjugação.

Popula as listas contendo os verbos e seus modelos.

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

Divide os dados em um treinamento e um conjunto de testes.

Parâmetros

- **threshold** – int. Tamanho mínimo da classe de conjugação a ser dividida.
- **proportion** – float. Proporção de amostras no conjunto de treino. Deve estar entre 0 e 1.

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

Bases:: class:‘ objeto‘

This class manages the scikit-learn pipeline.

O Pipeline inclui um vetorizador de recursos, um seletor de recursos e um classificador.

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.

Parâmetros

- **vectorizer** – scikit-learn Vectorizer
- **feature_selector** – classificador scikit-learn com um método fit_transform ()
- **classifier** – classificador scikit-learn com um método predict ()
- **language** – linguagem do corpus de verbos a ser analisado.

train (*samples*, *labels*)

Trains the pipeline on the supplied samples and labels.

Parâmetros

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

predict (*verbs*)

Prevê a classe de conjugação da lista de verbos fornecida.

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

Retorno lista. Lista de grupos de conjugação previstos.

4.2 Referência de API para as classes em mlconjug.PyVerbiste.py

PyVerbiste.

Uma biblioteca Python para conjugar verbos em francês, inglês, espanhol, italiano, português e romeno (mais em breve).

Ele contém dados de conjugação gerados por modelos de aprendizado de máquina usando a biblioteca python mlconjug.

Mais informações sobre mlconjug em https://pypi.org/project/mlconjug/

Os dados de conjugação estão em conformidade com o esquema XML definido pela Verbiste.
Mais informações sobre a Verbiste em https://perso.b2b2c.ca/~sarrazip/dev/conjug_manager.html

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

Parâmetros **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.

Parâmetros **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.

Parâmetros **conjugations_file** – string ou caminho do objeto. Caminho para o arquivo xml de conjugação.

_detect_allowed_endings()

Detecta os finais permitidos para verbos nos idiomas suportados.
Todos os idiomas suportados, exceto o inglês, restringem a forma que um verbo pode receber.
Como o inglês é muito mais produtivo e variado na morfologia de seus verbos, qualquer palavra é permitida como um verbo.

Retorno set. Um conjunto contendo as terminações permitidas de verbos no idioma de destino.

is_valid_verb(*verb*)

Verifica se o verbo é um verbo válido na língua dada.
Palavras inglesas são sempre tratadas como verbos possíveis.
Verbos em outros idiomas são filtrados por seus finais.

Parâmetros **verb** – string. O verbo para conjugar.

Retorno bool. Verdadeiro se o verbo é um verbo válido na língua. Falso caso contrário.

get_verb_info(*verb*)
Obtém informações verbais e retorna uma instância VerbInfo.

Parâmetros **verb** – string. Verb para conjugar.

Retorno Objeto VerbInfo ou Nenhum.

get_conjug_info(*template*)
Obtém informações de conjugação correspondentes ao modelo fornecido.

Parâmetros **template** – string. Nome do padrão final do verbo.

Retorno OrderedDict ou None. OrderedDict contendo os sufixos conjugados do template.

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

Esta é a classe que manipula os arquivos xml da Verbiste.

Parâmetros **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.

Parâmetros **verbs_file** – string ou caminho do objeto. Caminho para o arquivo xml verbos.

_parse_verbs(*file*)
Parses the XML file.

Parâmetros **file** – FileObject. Arquivo XML contendo os verbos.

Retorno OrderedDict. Um OrderedDict contendo o verbo e seu modelo para todos os verbos no arquivo.

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

Parâmetros **conjugations_file** – string ou caminho do objeto. Caminho para o arquivo xml de conjugação.

_parse_conjugations(*file*)
Parses the XML file.

Parâmetros **file** – FileObject. Arquivo XML contendo os modelos de conjugação.

Retorno OrderedDict. Um OrderedDict contendo todos os modelos de conjugação no arquivo.

_load_tense(*tense*)
Carregue e analise os formulários flexionados do tempo do arquivo xml.

Parâmetros **tense** – lista de tags xml contendo formulários flexionados. A lista de formulários flexionados para o tempo atual sendo processado.

Retorno list. List of inflected forms.

_detect_allowed_endings()

Detecta os finais permitidos para verbos nos idiomas suportados.
Todos os idiomas suportados, exceto o inglês, restringem a forma que um verbo pode receber.
Como o inglês é muito mais produtivo e variado na morfologia de seus verbos, qualquer palavra é permitida como um verbo.

Retorno set. Um conjunto contendo as terminações permitidas de verbos no idioma de destino.

get_conjug_info(*template*)
Obtém informações de conjugação correspondentes ao modelo fornecido.

Parâmetros **template** – string. Nome do padrão final do verbo.

Retorno OrderedDict ou None. OrderedDict contendo os sufixos conjugados do template.

get_verb_info(*verb*)
Obtém informações verbais e retorna uma instância VerbInfo.

Parâmetros **verb** – string. Verb para conjugar.

Retorno Objeto VerbInfo ou Nenhum.

is_valid_verb(*verb*)

Verifica se o verbo é um verbo válido na língua dada.

Palavras inglesas são sempre tratadas como verbos possíveis.
Verbos em outros idiomas são filtrados por seus finais.

Parâmetros **verb** – string. O verbo para conjugar.

Retorno bool. Verdadeiro se o verbo é um verbo válido na língua. Falso caso contrário.

class mlconjug.PyVerbiste.**VerbInfo**(*infinitive*, *root*, *template*)
Esta classe define a estrutura da informação verbal Verbiste.

Parâmetros

- **infinitive** – string. Forma infinitiva do verbo.
- **root** – string. Raiz lexical do verbo.
- **template** – string. Nome do padrão final do 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.

Parâmetros

- **verb_info** – Objeto VerbInfo»
- **conjug_info** – OrderedDict.
- **subject** – string. Alterna os pronomes abreviados ou completos. O valor padrão é "abrev". Selecione "pronome" para os pronomes completos.
- **predicted** – bool. Indica se as informações de conjugação foram previstas pelo modelo ou recuperadas do conjunto de dados.

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

_load_conjug()

Popula as formas flexionadas do verbo.
Esta é a versão genérica deste método.
Não acrescenta pronomes pessoais às formas conjugadas.
Este método pode manipular qualquer novo idioma se a estrutura de conjugação estiver em conformidade com o Esquema XML da Verbiste.

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

Esta classe define o Objeto Verbo Francês.

_load_conjug()

Popula as formas flexionadas do verbo.
Adiciona pronomes pessoais aos verbos flexionados.

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:: class:‘ mlconjug.PyVerbiste.Verb‘

Esta classe define o Inglês Objeto Verbo.

_load_conjug()

Popula as formas flexionadas do verbo.
Adiciona pronomes pessoais aos verbos flexionados.

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:: class:‘ mlconjug.PyVerbiste.Verb‘

Esta classe define o Objeto Verbo Espanhol.

_load_conjug()

Popula as formas flexionadas do verbo.
Adiciona pronomes pessoais aos verbos flexionados.

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:: class:‘ mlconjug.PyVerbiste.Verb‘

Esta classe define o Objeto Verbo Italiano.

_load_conjug()

Popula as formas flexionadas do verbo.
Adiciona pronomes pessoais aos verbos flexionados.

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:: class:‘ mlconjug.PyVerbiste.Verb‘

Esta classe define o Objeto Verbo Português.

_load_conjug()

Popula as formas flexionadas do verbo.
Adiciona pronomes pessoais aos verbos flexionados.

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:: class:‘ mlconjug.PyVerbiste.Verb‘

Esta classe define o Objeto Verbo Romeno.

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

_load_conjug()

Popula as formas flexionadas do verbo.
Adiciona pronomes pessoais aos verbos flexionados.

CAPÍTULO 5

Contribuindo

Contribuições são bem-vindas, e são muito apreciadas! Cada pequena ajuda, e crédito sempre será dado.

Você pode contribuir de várias maneiras:

5.1 Tipos de Contribuições

5.1.1 Reportar erros

Reportar bugs em https://github.com/SekouD/mlconjug/issues.

Se você está relatando um bug, por favor inclua:

- O nome e a versão do seu sistema operacional.
- Quaisquer detalhes sobre sua configuração local que possam ser úteis na solução de problemas.
- Etapas detalhadas para reproduzir o bug.

5.1.2 Corrigir erros

Olhe através dos problemas do GitHub para erros. Qualquer coisa marcada com » bug «e » help wanted «está aberta para quem quiser implementá-lo.

5.1.3 Implementar recursos

Examine os problemas do GitHub para recursos. Qualquer coisa marcada com » aprimoramento «e » ajuda desejada «está aberta a quem quiser implementá-la.

5.1.4 Write Documentation

O MLConjug poderia sempre usar mais documentação, seja como parte dos documentos oficiais do MLConjug, em docstrings, ou mesmo na web, em posts de blogs, artigos e outros.

5.1.5 Enviar feedback

A melhor maneira de enviar feedback é enviar um problema para https://github.com/SekouD/mlconjug/issues.

Se você está propondo um recurso:

- Explique em detalhes como isso funcionaria.
- Mantenha o escopo o mais estreito possível, para facilitar a implementação.
- Lembre-se que este é um projeto dirigido por voluntários e que as contribuições são bem-vindas :)

5.2 Iniciar!

Pronto para contribuir? Veja como configurar o "mlconjug" para desenvolvimento local.

1. Empurre o repositório‘ mlconjug‘ no GitHub.
2. Clone seu garfo localmente

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

3. Instale sua cópia local em um virtualenv. Assumindo que você tenha o virtualenvwrapper instalado, é assim que você configura seu fork para o desenvolvimento local:

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

4. Criar uma filial para desenvolvimento local

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

   Agora você pode fazer suas alterações localmente.

5. Quando você terminar de fazer alterações, verifique se suas alterações passam no flake8 e nos testes, incluindo testes de outras versões do Python com o tox

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

   Para obter o flake8 e o tox, basta instalá-los em seu virtualenv»

6. Confirme suas alterações e envie sua ramificação para o GitHub

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

7. Envie uma solicitação pull através do site do GitHub.

5.3 Diretrizes de solicitação pull

Antes de enviar uma solicitação de recebimento, verifique se ela atende a estas diretrizes:

1. O pedido pull deve incluir testes.
2. Se a solicitação pull adiciona funcionalidade, os documentos devem ser atualizados. Coloque sua nova funcionalidade em uma função com uma docstring e adicione o recurso à lista em 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 Dicas

Para executar um subconjunto de testes

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

CAPÍTULO 6

Créditos

6.1 Líder de desenvolvimento

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

6.2 Colaboradores

- O logotipo foi projetado por Zuur.

CAPÍTULO 7

História

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)

- Atualizado Anotações de tipo para toda a biblioteca para conformidade com o PEP-561.

7.10 3.1.1 (2018-06-26)

- Aperfeiçoamento de APIs menores (consulte Documentação da API)

7.11 3.1.0 (2018-06-24)

- Atualizados os modelos de conjugação para espanhol e português.
- Alterações internas no formato dos dados verbiste de xml para json para melhor manipulação de caracteres unicode.
- Nova classe ConjugManager para adicionar mais facilmente novos idiomas ao mlconjug.
- Aperfeiçoamento de APIs menores (consulte Documentação da API)

7.12 3.0.1 (2018-06-22)

- **Atualizado todos os modelos de previsão pré-treinados:**
  - Implementei um novo vectrorizer, extraindo recursos mais significativos.
  - Como resultado, o desempenho dos modelos passou pelo teto em todos os idiomas.
  - Recall e Precision estão intimamente próximos de 100%. Inglês sendo o único a alcançar uma pontuação perfeita em ambos Recall e Precision.
- **Principais alterações da API:**
  - Eu removi a classe EndingCustomVectorizer e refatorei sua funcionalidade em uma função de nível superior chamada extract_verb_features ()
  - O novo modelo aprimorado fornecido agora está sendo compactado com zip antes do lançamento, porque o espaço do recurso cresceu tanto que seu tamanho tornou impraticável a distribuição com o pacote.
  - Renomeado «Model.model» para «Model.pipeline»
  - Renomeado «DataSet.liste_verbes» e «DataSet.liste_templates» para «DataSet.verbs_list» e «DataSet.templates_list» respectivamente. (Perdoe meu francês ;-))
  - Adicionado os atributos «previsto» e «confidence_score» para a classe Verb.
  - Todo o pacote foi digitado como cheque. Em breve, adicionarei os stubs de tipo do mlconjug ao typeshed.

7.13 2.1.11 (2018-06-21)

- **Atualizado todos os modelos de previsão pré-treinados**
  - O Conjugador Francês tem precisão de cerca de 99,94% em predizer a classe de conjugação correta de um verbo francês. Esta é a linha de base, como eu tenho trabalhado nisso há algum tempo agora.
  - O Conjugador Inglês tem uma precisão de cerca de 99,78% na previsão da classe de conjugação correta de um verbo inglês. Este é um dos maiores avanços desde a versão 2.0.0
  - O Conjugador Espanhol tem uma precisão de cerca de 99,65% na previsão da classe de conjugação correta de um verbo espanhol. Ele também viu uma melhoria considerável desde a versão 2.0.0
  - O Conjugador Romeno tem uma precisão de cerca de 99,06% na previsão da classe de conjugação correta de um verbo romeno. Este é de longe o maior ganho. Eu modifiquei o vetorizador para melhor levar em conta as características morfológicas ou verbos romenos. (a pontuação anterior foi de cerca de 86%, por isso será bom para os nossos amigos romenos ter um conjugador de confiança)
  - O Conjugador Português tem uma precisão de cerca de 96,73% na previsão da classe correta de conjugação de um verbo português.
  - O Conjugador Italiano tem precisão de cerca de 94,05% em predizer a classe de conjugação correta de um verbo italiano.

7.14 2.1.9 (2018-06-21)

- **Agora o Conjugador adiciona informações adicionais ao objeto Verbo retornado.**

  - Se o verbo em consideração já está em Verbiste, a conjugação para o verbo é recuperada diretamente da memória.
  - Se o verbo sob consideração é desconhecido em Verbiste, a classe Conjugador agora configura o atributo booleano "predito" e a pontuação de confiança do atributo flutuante para a instância do objeto Verbo que o Conjugador.conjugado (verbo) retorna.

- Acrescentou Type annotations a toda a biblioteca para robustez e facilidade de dimensionamento.
- O desempenho dos modelos ingleses e romenos melhorou significativamente ultimamente. Eu acho que em mais algumas iterações eles estarão a par com o Modelo Francês que é o melhor desempenho no momento, já que eu tenho ajustado seus parâmetros para um par de anos agora. Não tanto com os outros idiomas, mas se você atualizar regularmente, verá ótimas melhorias no release 2.2.
- Melhorado a localização do programa.
- Agora a interface do usuário do mlconjug está disponível em francês, espanhol, italiano, português e romeno, além do inglês.
- Toda a documentação do projeto foi traduzida nos idiomas suportados.

7.15 2.1.5 (2018-06-15)

- Adicionado localização.
- Agora a interface do usuário do mlconjug está disponível em francês, espanhol, italiano, português e romeno, além do inglês.

7.16 2.1.2 (2018-06-15)

- Adicionada detecção de verbo inválida.

7.17 2.1.0 (2018-06-15)

- Atualizados todos os modelos de linguagem para compatibilidade com o scikit-learn 0.19.1.

7.18 2.0.0 (2018-06-14)

- Inclui modelo de conjugação em inglês.
- Inclui o modelo de conjugação em espanhol.
- Inclui o modelo de conjugação italiano.
- Inclui o modelo de conjugação em português.
- Inclui o modelo de conjugação romena.

7.19 1.2.0 (2018-06-12)

- Refatorou a API. Agora, é necessário um Conjugador de classe única para interagir com o módulo.
- Inclui melhor modelo de conjugação francesa.
- Adicionado suporte para vários idiomas.

7.20 1.1.0 (2018-06-11)

- Refatorou a API. Agora, é necessário um Conjugador de classe única para interagir com o módulo.
- Inclui melhor modelo de conjugação francesa.

7.21 1.0.0 (2018-06-10)

- Primeiro lançamento no PyPI.

CAPÍTULO 8

Índices e tabelas

- genindex
- modindex
- search

Índice de Módulos do Python

m

Índice

Símbolos

C

D

E

G

I

M

P

S

T

V