chana Documentation

Release 0.9

Jose Pereira

Aug 02, 2018

Contents

Welcome to Chana's documentation. Get started with *Installation* and then get an overview with the *Quickstart*. The rest of the docs describe each component of Chana in detail, with a full reference in the *API* section.

CHAPTER 1

User's Guide

This part of the documentation focuses on a quick set-up and instructions for using the basic NLP tools of Chana.

1.1 Installation

1.1.1 Python Version

We recommend using the version of Python 3. Chana supports Python 3.4 and newer.

1.1.2 Dependencies

To make chana work it is needed to have the following packages.

- NumPy the fundamental package for scientific computing with Python.
- Scikit-learn a package for machine learning in Python.
- Python-crfsuite a python binding to CRFsuite.

1.1.3 Install Chana

If you already have a working installation of numpy, scipy, and python-crfsuite the easiest way to install chana is using `pip`

```
pip install chana
```

1.2 Quickstart

This page gives a quick introduction to Chana. It assumes you already have Chana installed. If you do not, head over to the *Installation* section.

1.2.1 Using the Chana NLP Toolkit

A minimal code to use the Chana Toolkit looks something like this:

```
import chana.lemmatizer as lem
import chana.ner as ner
import chana.syllabificator as syl
import chana.pos_tagger as pos

lemmatizer = lem.ShipiboLemmatizer()
lemma = lemmatizer.lemmatize('pikanwe')

ship_ner = ner.ShipiboNER()
ner_tags = ship_ner.crf_tag('Enero Limanko atsa enra piawe')

tagger = pos.ShipiboPosTagger()
tags = tagger.pos_tag('Atsa enra piai')

syllables = syl.syllabify('atsabo')
```

So what did that code do?

1. First we imported the Chana tools (Lemmatizer, NER, Syllabificator and Pos-Tagger).
2. Next we create an instance of the Shipibo Lemmatizer and then we used it to get the lemma of a Shipibo word.
3. We then create an instance of the Shipibo NER and use it to get the NER tags of a Shipibo sentence.
4. Next we create an instance of the Shipibo Pos-Tagger and use it to get the pos-tags of a Shipibo sentence.
5. Finally we use the syllabify function of the Shipibo Syllabificator in order to get the syllables of a Shipibo word.

Just save it as `test.py` or something similar, to start using the chana tools with your own input.

CHAPTER 2

API Reference

Information about the functions, classes or methods of Chana.

2.1 API

This part of the documentation covers all the functions of Chana.

2.1.1 Modules

Lemmatizer

Lemmatizer for shipibo-konibo Source model is from the Chana project and a use KNeighborsClassifier from scikit-learn

class chana.lemmatizer.**GeneralLemmatizer**(*features_length=10*, *n_neighbors=5*)

Instance of a new lemmatizer to be trained and used

get_lemma(*rule*, *word*)

Method that returns the lemma of a word given a possible rule

Parameters

- **rule** (*list*) – a rule to transform a word
- **word** (*str*) – a word to be transformed

Returns word transformed

Return type str

Example

```
>>> import chana.lemmatizer
>>> lemmatizer = chana.lemmatizer.GeneralLemmatizer()
>>> lemmatizer.get_lemma(['bo>'],'shipibobo')
'shipibo'
```

get_rule(*word*)

Method that returns the transformation rule for a word

Parameters **word** (*str*) – a word to get the rule

Returns numpy array with the rule

Return type array

Example

```
>>> import chana.lemmatizer
>>> lemmatizer = chana.lemmatizer.GeneralLemmatizer()
>>> lemmatizer.get_rule('perrito')
array(['ito>0'], dtype='<U16')
```

lemmatize(*word*)

Method that predicts the lemma of a word with the trained model

Parameters **word** (*str*) – a word to get the lemma

Returns lemma of the word

Return type str

Example

```
>>> import chana.lemmatizer
>>> lemmatizer = chana.lemmatizer.GeneralLemmatizer()
>>> lemmatizer.lemmatize('perrito')
'perro'
```

preprocess_word(*word*)

Method that turns a word in an array of features for the classifier according to its features_length

Parameters **word** (*str*) – a word to be transformed

Returns list with the features

Return type list

Example

```
>>> import chana.lemmatizer
>>> lemmatizer = chana.lemmatizer.GeneralLemmatizer()
>>> lemmatizer.preprocess_word('perritos')
[115, 111, 116, 105, 114, 114, 101, 112, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

train(*words*, *lemmas*)

Method that trains a new lemmatizer with a list of words and a list of lemmas of the same size

Parameters

- **words** (*list*) – list of words
- **lemmas** (*list*) – list of lemmas

Returns none

Return type None

Example

```
>>> import chana.lemmatizer
>>> lemmatizer = chana.lemmatizer.GeneralLemmatizer()
>>> lemmas = ['perro','gato','mono']
>>> words = ['perritos','gatitos','monotes']
>>> lemmatizer.train(words,lemmas)
```

class chana.lemmatizer.**ShipiboLemmatizer**

Instance of the pre-trained shipibo lemmatizer

get_lemma(*rule*, *word*)

Method that returns the lemma of a shipibo word given a possible rule

Parameters

- **rule** (*list*) – a rule to transform a word
- **word** (*str*) – a word to be transformed

Returns word transformed

Return type str

Example

```
>>> import chana.lemmatizer
>>> lemmatizer = chana.lemmatizer.ShipiboLemmatizer()
>>> lemmatizer.get_lemma(['bo>'],'shipibobo')
'shipibo'
```

get_rule(*word*)

Method that returns the transformation rule for a shipibo word

Parameters **word** (*str*) – a word to get the rule

Returns numpy array with the rule

Return type array

Example

```
>>> import chana.lemmatizer
>>> lemmatizer = chana.lemmatizer.ShipiboLemmatizer()
>>> lemmatizer.get_rule('pikanwe')
array(['anwe>i'], dtype='<U16')
```

lemmatize(*word*)

Method that predicts the lemma of a shipibo word

Parameters **word** (*str*) – a word to get the lemma

Returns lemma of the word

Return type str

Example

```
>>> import chana.lemmatizer
>>> lemmatizer = chana.lemmatizer.ShipiboLemmatizer()
>>> lemmatizer.lemmatize('pikanwe')
'piki'
```

preprocess_word(*word*)

Method that turns a word in an array of features for the classifier

Parameters **word** (*str*) – a word to be transformed

Returns list with the features

Return type list

Example

```
>>> import chana.lemmatizer
>>> lemmatizer = chana.lemmatizer.ShipiboLemmatizer()
>>> lemmatizer.preprocess_word('shipibobo')
[111, 98, 111, 98, 105, 112, 105, 104, 115, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

chana.lemmatizer.**has_shipibo_suffix**(*str*)

Function that returns the possible existence of a shipo suffix in a a word

Parameters **str** (*str*) – word to evaluate

Returns True or False

Return type bool

Example

```
>>> import chana.lemmatizer
>>> chana.lemmatizer.has_shipibo_suffix('pianra')
True
```

chana.lemmatizer.**longest_common_substring**(*string1*, *string2*)

Function to find the longest common substring of two strings

Parameters

- **string1** (*str*) – string1
- **string2** (*str*) – string2

Returns longest common substring

Return type str

Example

```
>>> import chana.lemmatizer
>>> chana.lemmatizer.longest_common_substring('limanko','limanra')
'liman'
```

chana.lemmatizer.**replace_last**(*source_string*, *replace_what*, *replace_with*)

Function that replaces the last ocurrence of a string in a word

Parameters

- **source_string** (*str*) – the source string
- **replace_what** (*str*) – the substring to be replaced
- **replace_with** (*str*) – the string to be inserted

Returns string with the replacement

Return type str

Example

```
>>> import chana.lemmatizer
>>> chana.lemmatizer.replace_last('piati','ti','ra')
'piara'
```

chana.lemmatizer.**shipibo_suffixes**()
Function that returns a list with all the shipibo suffixes

Returns list with all the suffixes

Return type list

Example

```
>>> import chana.lemmatizer
>>> chana.lemmatizer.shipibo_suffixes()
['naan', 'yama', 'men', 'iosma', ..., 'shoko']
```

NER

Named-entity recognizer for shipibo-konibo Source model is from the Chana project and use predefined rules for the language as well as a crf from pycrfsuite

class chana.ner.**ShipiboNER**
Instance of the rule based NER for shipibo

check_dates(*words*, *entity_tag*)
Inner method that tags the dates of a sentence with 'FEC'

Parameters

- **words** (*list*) – a list of words to be evaluated
- **entity_tag** (*list*) – a list of words to be evaluated

Returns none

Return type None

check_locations(*words*, *entity_tag*)
Inner method that tags the locations of a sentence with 'LOC'

Parameters

- **words** (*list*) – a list of words to be evaluated
- **entity_tag** (*list*) – a list of words to be evaluated

Returns none

Return type None

check_names(*words*, *entity_tag*)
Inner method that tags the names/persons of a sentence with 'PER'

Parameters

- **words** (*list*) – a list of words to be evaluated
- **entity_tag** (*list*) – a list of words to be evaluated

Returns none

Return type None

check_numbers(*words*, *entity_tag*)
Inner method that tags the numbers of a sentence with 'NUM'

Parameters

- **words** (*list*) – a list of words to be evaluated
- **entity_tag** (*list*) – a list of words to be evaluated

Returns none

Return type None

check_organizations(*words*, *entity_tag*)
Inner method that tags the organizations of a sentence with 'ORG'

Parameters

- **words** (*list*) – a list of words to be evaluated
- **entity_tag** (*list*) – a list of words to be evaluated

Returns none

Return type None

crf_tag(*sentence*)
Method that tags a sentence with the rule based method and then with the crf model

Parameters **sentence** (*str*) – a sentence to be evaluated

Returns list with the ner tags

Return type list

Example

```
>>> import chana.ner
>>> ner = chana.ner.ShipiboNer()
>>> ner.crf_tag('Limanko enra atsawe')
['LOC', 'O', 'O']
```

rule_tag(*sentence*)
Method that tags a sentence with the rule based system

Parameters **sentence** (*str*) – a sentence to be evaluated

Returns list with the ner tags

Return type list

Example

```
>>> import chana.ner
>>> ner = chana.ner.ShipiboNer()
>>> ner.rule_tag('Limanko enra atsawe')
['LOC', 'O', 'O']
```

sent2features(*sent*)
Inner method that add features to a sentence to be tagged by the crf model

Parameters **sent** (*list*) – a sentence in list form to be transformed into features

Returns list with features

Return type list

word2features(*sent*, *i*)
Inner method that add features to the words of a sentence to be tagged by the crf model

Parameters

- **sent** (*list*) – a sentence in list form to be transformed into features
- **i** (*int*) – index of the word to be evaluated

Returns list with the features for the indexed word

Return type list

chana.ner.**is_date**(*word*)
Function that returns 'FEC' if a shipo word is a date or False if not

Parameters **word** (*str*) – a word to be evaluated

Returns 'FEC' if a shipo word is a date or False if not

Return type str

Example

```
>>> import chana.ner
>>> chana.ner.is_date('Agosto')
'FEC'
```

chana.ner.**is_location**(*word*)
Function that returns 'LOC' if a shipo word is a location or False if not

Parameters **word** (*str*) – a word to be evaluated

Returns 'LOC' if a shipo word is a location or False if not

Return type str

Example

```
>>> import chana.ner
>>> chana.is_location.is_name('Limanko')
'LOC'
```

chana.ner.**is_name**(*word*)
Function that returns 'PER' if a shipo word is a proper name/person or False if not

Parameters **word** (*str*) – a word to be evaluated

Returns 'PER' if a shipo word is a proper name/person or False if not

Return type str

Example

```
>>> import chana.ner
>>> chana.ner.is_name('Adriano')
'PER'
```

chana.ner.**is_number**(*word*)
Function that returns 'NUM' if a shipo word is a number or False if not

Parameters **word** (*str*) – a word to be evaluated

Returns 'NUM' if a shipo word is a number or False if not

Return type str

Example

```
>>> import chana.ner
>>> chana.ner.is_number('kimisha')
'NUM'
```

chana.ner.**is_organization**(*word*)

Function that returns 'ORG' if a shipo word is an organization or False if not

Parameters **word** (*str*) – a word to be evaluated

Returns 'ORG' if a shipo word is an organization or False if not

Return type str

Example

```
>>> import chana.ner
>>> chana.ner.is_organization('AUT')
'ORG'
```

chana.ner.**load_array**(*file*, *array*)

Inner function that loads the information of a file into a list

Parameters

- **file** (*File*) – a file to be loaded
- **array** (*list*) – a list to be populated with the information from the file

Returns none

Return type None

Pos_Tagger

Part-of-Speech (POS) Tagger for shipibo-konibo. Source model is from the Chana project

class chana.pos_tagger.**ShipiboPosTagger**

Instance of the pre-trained shipibo part-of-speech tagger

features(*sentence*, *tags*, *index*)

Method that returns the features of a word in a sentence to be used by the model

Parameters

- **sentence** (*str*) – a sentence in shipibo-konibo
- **tags** (*list*) – tags to be returned for the word
- **index** (*int*) – position of the word in the sentence

Returns dict of features for the indexed word

Return type dict

Example

```
>>> import chana.pos_tagger
>>> tagger = chana.pos_tagger.ShipiboPosTagger()
>>> tagger.features('Atsa ea piai',['','',''],2)
{'word': 's', 'prevWord': 't', 'nextWord': 'a', 'isFirst': False, 'isLast':␣
↪False, 'isCapitalized': False, 'isAllCaps': False, 'isAllLowers': True,
↪'prefix-1': 's', 'prefix-2': 's', 'prefix-3': 's', 'prefix-4': 's', 'suffix-
↪1': 's', 'suffix-2': 's', 'suffix-3': 's', 'suffix-4': 's', 'tag-1': '',
↪'tag-2': ''}
```

(continues on next page)

(continued from previous page)

full_pos_tag(*sentence*)

Method that predict the pos-tags of a shipibo sentence and returns the full tag in spanish

Parameters **sentence** (*str*) – a sentence in shipibo-konibo

Returns list of the tags in spanish

Return type list

Example

```
>>> import chana.pos_tagger
>>> tagger = chana.pos_tagger.ShipiboPosTagger()
>>> tagger.full_pos_tag('Atsa ea piai')
['Nombre', 'Pronombre', 'Verbo']
```

get_complete_tag(*pos*)

Method that returns the full tag in spanish of a tag

Parameters **pos** (*str*) – a pos tag in the UD format

Returns str with the tag in spanish

Return type str

Example

```
>>> import chana.pos_tagger
>>> tagger = chana.pos_tagger.ShipiboPosTagger()
>>> tagger.get_complete_tag('ADJ')
'Adjetivo'
```

pos_tag(*sentence*)

Method that predict the pos-tags of a shipibo sentence in the UD format

Parameters **sentence** (*str*) – a sentence in shipibo-konibo

Returns list of the tags in UD format

Return type list

Example

```
>>> import chana.pos_tagger
>>> tagger = chana.pos_tagger.ShipiboPosTagger()
>>> tagger.pos_tag('Atsa ea piai')
['NOUN', 'PRON', 'VERB']
```

Syllabificator

Syllabificator for shipibo-konibo. General functions and rules to syllabify a shipibo-konibo word

chana.syllabificator.**accentuate**(*letter*)

Function that adds the accentuation mark of a letter:

Parameters **letter** (*str*) – a letter to be accentuated

Returns letter accentuated

Return type str

Example

```
>>> import chana.syllabificator
>>> chana.syllabificator.accentuate('a')
á
```

chana.syllabificator.**change**(*syllable*)

Function that returns the original form of a syllable

Parameters **syllable** (*str*) – a syllable to be transformed

Returns syllable with its original form

Return type str

Example

```
>>> import chana.syllabificator
>>> chana.syllabificator.change('1a')
cha
```

chana.syllabificator.**get_vc**(*word*)

Function that returns all the vowels and consonants of a word

Parameters **word** (*str*) – word to get its vowels and consonants

Returns list of 'V' and 'C' for each letter of the word

Return type list

Example

```
>>> import chana.syllabificator
>>> chana.syllabificator.get_vc('piti')
[['p', 'C'], ['i', 'V'], ['t', 'C'], ['i', 'V']]
```

chana.syllabificator.**syllabify**(*word*)

Function that returns all the syllables of a word

Parameters **word** (*str*) – a word to get its syllables

Returns list of syllables

Return type list

Example

```
>>> import chana.syllabificator
>>> chana.syllabificator.syllabify('atsabo')
['a', 'tsa', 'bo']
```

CHAPTER 3

Additional Notes

Legal information and changelog.

3.1 Chana Changelog

3.1.1 Version 0.9

First public release.

3.2 License

Chana is licensed under a three clause MIT License. It basically means: A permissive license that lets people do anything they want with the code as long as they provide attribution back to the owner and don't hold the owner liable. The full license text can be found below (*Chana License*).

3.2.1 Authors

Chana is written and maintained by Jose Pereira and various contributors:

Development Lead

Jose Pereira <jpereira@pucp.edu.pe>

Contributors

- Jose Pereira (Lemmatizer)
- Rodolfo Mercado (Pos-Tagger)

- Vivian Gongora (NER)
- Carlo Alva (Syllabificator)

3.2.2 General License Definitions

The following section contains the full license texts for Flask and the documentation.

- "AUTHORS" hereby refers to all the authors listed in the *Authors* section.
- The "*Chana License*" applies to all the source code shipped as part of Chana as well as documentation.

3.2.3 Chana License

Index

W