Etiquetado en NLTK

Pipeline básico para Ingles

import nltk  
nltk.download('punkt')  
nltk.download('averaged\_perceptron\_tagger')  
from nltk import word\_tokenize

text = word\_tokenize("And now here I am enjoying today")  
nltk.pos\_tag(text)

[('And', 'CC'),

('now', 'RB'),

('here', 'RB'),

('I', 'PRP'),

('am', 'VBP'),

('enjoying', 'VBG'),

('today', 'NN')]

#@title Categoria gramatical de cada etiqueta  
nltk.download('tagsets')  
for tag in ['CC', 'RB', 'PRP', 'VBP', 'VBG', 'NN']:  
   print(nltk.help.upenn\_tagset(tag))

Categoria gramatical de cada etiqueta

[nltk\_data] Downloading package tagsets to /root/nltk\_data...

[nltk\_data] Package tagsets is already up-to-date!

CC: conjunction, coordinating

& 'n and both but either et for less minus neither nor or plus so

therefore times v. versus vs. whether yet

None

RB: adverb

occasionally unabatingly maddeningly adventurously professedly

stirringly prominently technologically magisterially predominately

swiftly fiscally pitilessly ...

None

PRP: pronoun, personal

hers herself him himself hisself it itself me myself one oneself ours

ourselves ownself self she thee theirs them themselves they thou thy us

None

VBP: verb, present tense, not 3rd person singular

predominate wrap resort sue twist spill cure lengthen brush terminate

appear tend stray glisten obtain comprise detest tease attract

emphasize mold postpone sever return wag ...

None

text = word\_tokenize("They do not permit other people to get residence permit")  
nltk.pos\_tag(text)

Palabras homónimas

[('They', 'PRP'),

('do', 'VBP'),

('not', 'RB'),

('permit', 'VB'),

('other', 'JJ'),

('people', 'NNS'),

('to', 'TO'),

('get', 'VB'),

('residence', 'NN'),

('permit', 'NN')]

Etiquetado en Español

Para el ingles, NLTK tiene tokenizador y etiquetador pre-entrenados por defecto. En cambio, para otros idiomas es preciso entrenarlo previamente.

* usamos el corpus cess\_esp <https://mailman.uib.no/public/corpora/2007-October/005448.html>
* el cual usa una convención de etiquetas gramaticales dada por el grupo EAGLES <https://www.cs.upc.edu/~nlp/tools/parole-sp.html>

nltk.download('cess\_esp')  
from nltk.corpus import cess\_esp as cess  
from nltk import UnigramTagger as ut  
from nltk import BigramTagger as bt

[nltk\_data] Downloading package cess\_esp to /root/nltk\_data...

[nltk\_data] Unzipping corpora/cess\_esp.zip.

#@title Entrenamiendo del tagger por unigramas  
fraction = int(len(cess\_sents)\*90/100)  
cess\_sents = cess.tagged\_sents()  
uni\_tagger = ut(cess\_sents[:fraction])  
uni\_tagger.evaluate(cess\_sents[fraction+1:])

Entrenamiendo del tagger por unigramas

0.8069484240687679

uni\_tagger.tag("Yo soy una persona muy amable".split(" "))

[('Yo', 'pp1csn00'),

('soy', 'vsip1s0'),

('una', 'di0fs0'),

('persona', 'ncfs000'),

('muy', 'rg'),

('amable', 'aq0cs0')]

#@title Entrenamiento del tagger por bigramas  
fraction = int(len(cess\_sents)\*90/100)  
bi\_tagger = bt(cess\_sents[:fraction])  
bi\_tagger.evaluate(cess\_sents[fraction+1:])

Entrenamiento del tagger por bigramas

0.1095272206303725

bi\_tagger.tag("Yo soy una persona muy amable".split(" "))

[('Yo', 'pp1csn00'),

('soy', 'vsip1s0'),

('una', None),

('persona', None),

('muy', None),

('amable', None)]

Etiquetado mejorado con Stanza (StanfordNLP)

**¿Que es Stanza?**

* El grupo de investigacion en NLP de Stanford tenía una suite de librerias que ejecutaban varias tareas de NLP, esta suite se unifico en un solo servicio que llamaron **CoreNLP** con base en codigo java: <https://stanfordnlp.github.io/CoreNLP/index.html>
* Para python existe **StanfordNLP**: <https://stanfordnlp.github.io/stanfordnlp/index.html>
* Sin embargo, **StanfordNLP** ha sido deprecado y las nuevas versiones de la suite de NLP reciben mantenimiento bajo el nombre de **Stanza**: <https://stanfordnlp.github.io/stanza/>

Stanza está construido con componentes de red neuronal de alta precisión que también permiten un entrenamiento y una evaluación eficientes con sus propios datos anotados. Los módulos están construidos sobre la biblioteca de PyTorch . Obtendrá un rendimiento mucho más rápido si ejecuta este sistema en una máquina habilitada para GPU

!pip install stanza

Collecting stanza

Downloading <https://files.pythonhosted.org/packages/27/9c/60689521a971a57dd02d2925105efedefa9dccd76c9a0b92566683d43e89/stanza-1.0.1-py3-none-any.whl> (193kB)

|████████████████████████████████| 194kB 2.8MB/s

Requirement already satisfied: torch>=1.3.0 in /usr/local/lib/python3.6/dist-packages (from stanza) (1.5.1+cu101)

Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from stanza) (4.41.1)

Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests->stanza) (1.24.3)

Installing collected packages: stanza

Successfully installed stanza-1.0.1

import stanza  
stanza.download('es')

Downloading <https://raw.githubusercontent.com/stanfordnlp/stanza-resources/master/resources_1.0.0.json>: 115kB [00:00, 10.1MB/s]

nlp = stanza.Pipeline('es', processors='tokenize,pos')  
doc = nlp('yo soy una persona muy amable')

2020-07-13 01:53:05 INFO: Loading these models for language: es (Spanish):

=======================

| Processor | Package |

-----------------------

| tokenize | ancora |

| pos | ancora |

=======================

2020-07-13 01:53:06 INFO: Use device: cpu

2020-07-13 01:53:06 INFO: Loading: tokenize

2020-07-13 01:53:06 INFO: Loading: pos

2020-07-13 01:53:07 INFO: Done loading processors!

for sentence in doc.sentences:

  for word in sentence.words:

    print(word.text, word.pos)

yo PRON

soy AUX

una DET

persona NOUN

muy ADV

amable ADJ

Referencias adicionales:

* Etiquetado POS con Stanza <https://stanfordnlp.github.io/stanza/pos.html#accessing-pos-and-morphological-feature-for-word>
* Stanza | Github: <https://github.com/stanfordnlp/stanza>
* Articulo en ArXiv: <https://arxiv.org/pdf/2003.07082.pdf>

stanza también tiene una demo en linea: <http://stanza.run/>