



INSTITUTO POLITÉCNICO NACIONAL  
CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

**ALUMNO:**

Guadalupe Javier Armenta García

**MATERIA:**

Tratamiento de lenguaje natural

**PROGRAMA:**

Maestría en Ciencias de la Computación

**TAREA:**

FreeLing y librería Stanza

**FECHA:**

28 de Abril del 2020

## Stanza

Es un paquete de análisis de lenguaje natural de Python. Contiene herramientas, que se pueden usar en una tubería, para convertir una cadena de texto en lenguaje humano en listas de oraciones y palabras. Sirve para:

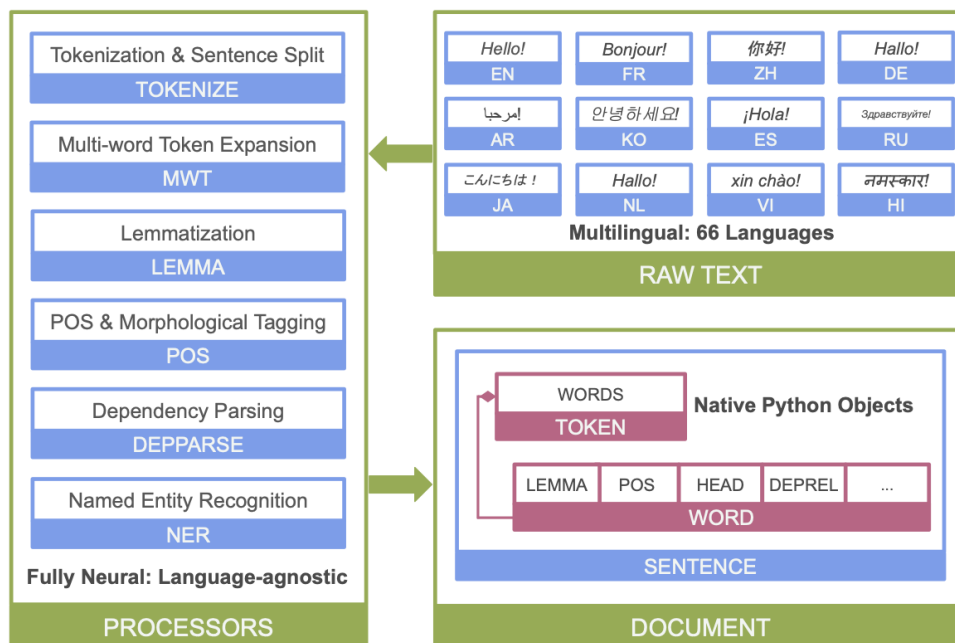
- Generar formas básicas de esas palabras
- Sus partes del discurso y características morfológicas
- Para dar un análisis de dependencia de estructura sintáctica
- Para reconocer entidades con nombre.

El kit de herramientas está diseñado para ser paralelo entre más de 70 idiomas, utilizando el *Universal Dependencies formalism*.

### Características:

1. Implementación nativa de Python que requiere esfuerzos mínimos para configurar
2. Full neural network pipeline para análisis de texto robusto, que incluye tokenization, multi-word token (MWT) expansion, lemmatization, part-of-speech (POS) y morphological features tagging, dependency parsing y named entity recognition
3. Modelos neuronales entrenados que admiten 66 lenguajes (humanos);
4. Una interfaz Python estable y oficialmente mantenida para CoreNLP.

Descripción general de neural network NLP pipeline de Stanza:



## INSTALACION DE STANZA

Para ello se usó del instalador de paquetes de python 'pip', ejecutando en el siguiente comando.

```
pip install stanza
```

## EJECUCION DE STANZA

Una vez instalado Stanza se descargó el paquete de idioma en inglés y se ejecutó el primer pipeline para probar su correcta instalación. (El código es ejecutado sobre Anaconda).

```
In [4]: import stanza
stanza.download('en') # download the english model
nlp = stanza.Pipeline('en') # initialize English neural pipeline
doc = nlp("Barack Obama was born in Hawaii.") # run annotation over a sentence
```

Debido a que ya se había descargado el paquete en inglés indicará que existía previamente.

```
Downloading https://raw.githubusercontent.com/stanfordnlp/stanza-resources/master/resources_1.0.0.json: 116kB [00:00, 1.76MB/s]
2020-04-28 12:59:12 INFO: Downloading default packages for language: en (English)...
2020-04-28 12:59:13 INFO: File exists: /home/javier/stanza_resources/en/default.zip.
2020-04-28 12:59:19 INFO: Finished downloading models and saved to /home/javier/stanza_resources.
2020-04-28 12:59:19 INFO: Loading these models for language: en (English):
```

Processor	Package
tokenize	ewt
pos	ewt
lemma	ewt
depparse	ewt
ner	ontonotes

```
2020-04-28 12:59:19 INFO: Use device: cpu
2020-04-28 12:59:19 INFO: Loading: tokenize
2020-04-28 12:59:19 INFO: Loading: pos
2020-04-28 12:59:20 INFO: Loading: lemma
2020-04-28 12:59:20 INFO: Loading: depparse
2020-04-28 12:59:21 INFO: Loading: ner
2020-04-28 12:59:21 INFO: Done loading processors!
```

## PROCCESORS

- Tokenization

```
import stanza

nlp = stanza.Pipeline(lang='en', processors='tokenize') # args: lang in english, processor to be used= Tokenizer
doc = nlp('Javier is excited to learn more about stanza')
for i, sentence in enumerate(doc.sentences):
    print(f'==== Sentence {i+1} tokens =====')
    print(*[f'id: {token.id}\ttext: {token.text}' for token in sentence.tokens], sep='\n')
```

2020-04-28 13:08:36 INFO: Loading these models for language: en (English):

Processor	Package
tokenize	ewt

2020-04-28 13:08:36 INFO: Use device: cpu  
2020-04-28 13:08:36 INFO: Loading: tokenize  
2020-04-28 13:08:36 INFO: Done loading processors!

==== Sentence 1 tokens =====

```
id: 1 text: Javier
id: 2 text: is
id: 3 text: excited
id: 4 text: to
id: 5 text: learn
id: 6 text: more
id: 7 text: about
id: 8 text: stanza
```

- Morphological features tagging

```
nlp = stanza.Pipeline(lang='en', processors='tokenize,mwt,pos') # applying tokenizer, POSprocessor and MWTpro..
##Running the POSProcessor requires the TokenizeProcessor and MWTProcessor.
doc = nlp('Javier is excited to learn more about stanza')
print(*[f'word: {word.text}\tupos: {word.upos}\txpos: {word.xpos}\tfeats: {word.feats if word.feats else "_"}'
        for sent in doc.sentences for word in sent.words], sep='\n')
```

2020-04-28 13:13:14 WARNING: Can not find mwt: default from official model list. Ignoring it.  
2020-04-28 13:13:14 INFO: Loading these models for language: en (English):

Processor	Package
tokenize	ewt
pos	ewt

2020-04-28 13:13:14 INFO: Use device: cpu  
2020-04-28 13:13:14 INFO: Loading: tokenize  
2020-04-28 13:13:14 INFO: Loading: pos  
2020-04-28 13:13:15 INFO: Done loading processors!

word: Javier	upos: PROP	xpos: NNP	feats: Number=Sing
word: is	upos: AUX	xpos: VBZ	feats: Mood=Ind Number=Sing Person=3 Tense=Pres VerbForm=Fin
word: excited	upos: ADJ	xpos: JJ	feats: Degree=Pos
word: to	upos: PART	xpos: TO	feats: _
word: learn	upos: VERB	xpos: VB	feats: VerbForm=Inf
word: more	upos: ADV	xpos: RBR	feats: _
word: about	upos: ADP	xpos: IN	feats: _
word: stanza	upos: NOUN	xpos: NN	feats: Number=Sing

Como podemos ver ha clasificado cada token de acuerdo a la nomenclatura EAGLES.

- Lemmatization

```

: nlp = stanza.Pipeline(lang='en', processors='tokenize,mwt,pos,lemma') #processors needed for lemmatization
doc = nlp('Javier has been very excited for applying lemmatization')
print(*[f'word: {word.text+" "}\tlemma: {word.lemma}' for sent in doc.sentences for word in sent.words], sep='\n')

```

2020-04-28 13:21:57 WARNING: Can not find mwt: default from official model list. Ignoring it.  
2020-04-28 13:21:57 INFO: Loading these models for language: en (English):

Processor	Package
tokenize	ewt
pos	ewt
lemma	ewt

2020-04-28 13:21:57 INFO: Use device: cpu  
2020-04-28 13:21:57 INFO: Loading: tokenize  
2020-04-28 13:21:57 INFO: Loading: pos  
2020-04-28 13:21:58 INFO: Loading: lemma  
2020-04-28 13:21:58 INFO: Done loading processors!

```

word: Javier      lemma: Javier
word: has         lemma: have
word: been        lemma: be
word: very        lemma: very
word: excited     lemma: excited
word: for         lemma: for
word: applying    lemma: apply
word: lemmatization lemma: lemmatization

```

- Dependency parsing

```

nlp = stanza.Pipeline(lang='en', processors='tokenize,mwt,pos,lemma,depparse')
doc = nlp('This processor creates a syntactic tree')
print(*[f'id: {word.id}\tword: {word.text}\thead id: {word.head}\thead: {sent.words[word.head-1].text if word.head >
for sent in doc.sentences for word in sent.words], sep='\n')

```

id: 1	word: This	head id: 2	head: processor	deprel: det
id: 2	word: processor	head id: 3	head: creates	deprel: nsubj
id: 3	word: creates	head id: 0	head: root	deprel: root
id: 4	word: a	head id: 6	head: tree	deprel: det
id: 5	word: syntactic	head id: 6	head: tree	deprel: amod
id: 6	word: tree	head id: 3	head: creates	deprel: obj

## FREELING

FreeLing es una biblioteca de C ++ que proporciona funcionalidades de análisis de lenguaje (morphological analysis, named entity detection, PoS-tagging, parsing, Word Sense Disambiguation, Semantic Role Labelling, etc.) para una variedad de idiomas.

Su sitio web contiene una demo que permite interactuar con ella para realizar dichas tareas.

Como bien sabemos para la ejecución de ciertas tareas es necesario ejecutar tareas previas, el demo de Freeling no contiene la tarea de tokenization ni lemmatization por lo cual introduciremos todas juntas en el análisis morfológico.

- Tokenization, morphological analysis y lemmatization

## FreeLing 4.1 - An Open-Source Suite of Language Analyzers

*Nothing more than FreeLing*

### Write your sentences

Speaking at a news conference in Kiev, the capital, Mr. Poroshenko said that he planned to meet with Mr. Putin within the next three weeks and expressed confidence that the cease-fire with pro-Russian rebels would hold.

### Analysis options

- ☒ Number recognition
- ☒ Date/Time recognition
- ☒ Quantities, ratios, and percentages
- ☒ Named Entity detection
- ☐ Named Entity classification
- ☒ Multiword detection
- ☐ Phonetic encoding
- ☐ No sense annotation
- ☐ WN sense annotation: All senses
- ☐ WN sense annotation: [UKB](#) disambiguation

### Select language

English ▼

### Select output

Morphological Analysis ▼

Submit

### ▼ Sentences

#### Sentence 1

Speaking	at	a	news	conference	in	Kiev	,	the	capital	,	Mr. Poroshenko	said	that
speaking VBG 1	at IN 0.999978	a DT 0.998827	news NN 1	conference NN 1	in IN 0.987107	kiev NP 1	,	the DT 1	capital NN 0.974929	,	mr_poroshenko NP 1	say VBD 0.986601	that IN 0.539936
	at RP 2.16113e-05	a NN 0.00061172			in RP 0.0100582				capital JJ 0.0250706			say VBN 0.0133992	that WDT 0.277518
		1 Z 0.000561164			in RB 0.00283508								that DT 0.182522
													that RB 1.20707e-05
													that WP 1.20707e-05

he	planned	to	meet	with	Mr. Putin	within	the	next	three	weeks	and	expressed
he PRP 1	plan VBN 0.719561	to TO 0.997897	meet VB 0.938091	with IN 0.999515	mr_putin NP 1	within IN 0.999495	the DT 1	next JJ 0.857356	3 Z 0.999843	week NNS 1	and CC 1	express VBD 0.574627
	plan VBD 0.280439	to IN 0.0021026	meet VBP 0.061049	with RP 0.000484966		within RB 0.000504541		next IN 0.13255	three DT 0.000157183			express VBN 0.425373
			meet NN 0.000859845					next RB 0.0100943				

Como podemos observar en la imagen anterior la demo tiene como salida el conjunto de tokens de color negro con su lema de color azul (said -> say) y su información morfológica en formato EAGLE

- Full parsing y árbol sintáctico

Al ejecutar esta tarea en la demo arroja como resultado todos los procesamiento anteriores además de indicar el árbol sintáctico asociado a dicho texto.

Sentence 1

Speaking at a news conference in Kiev, the capital, Mr. Poroshenko said that he planned to meet with Mr. Putin

