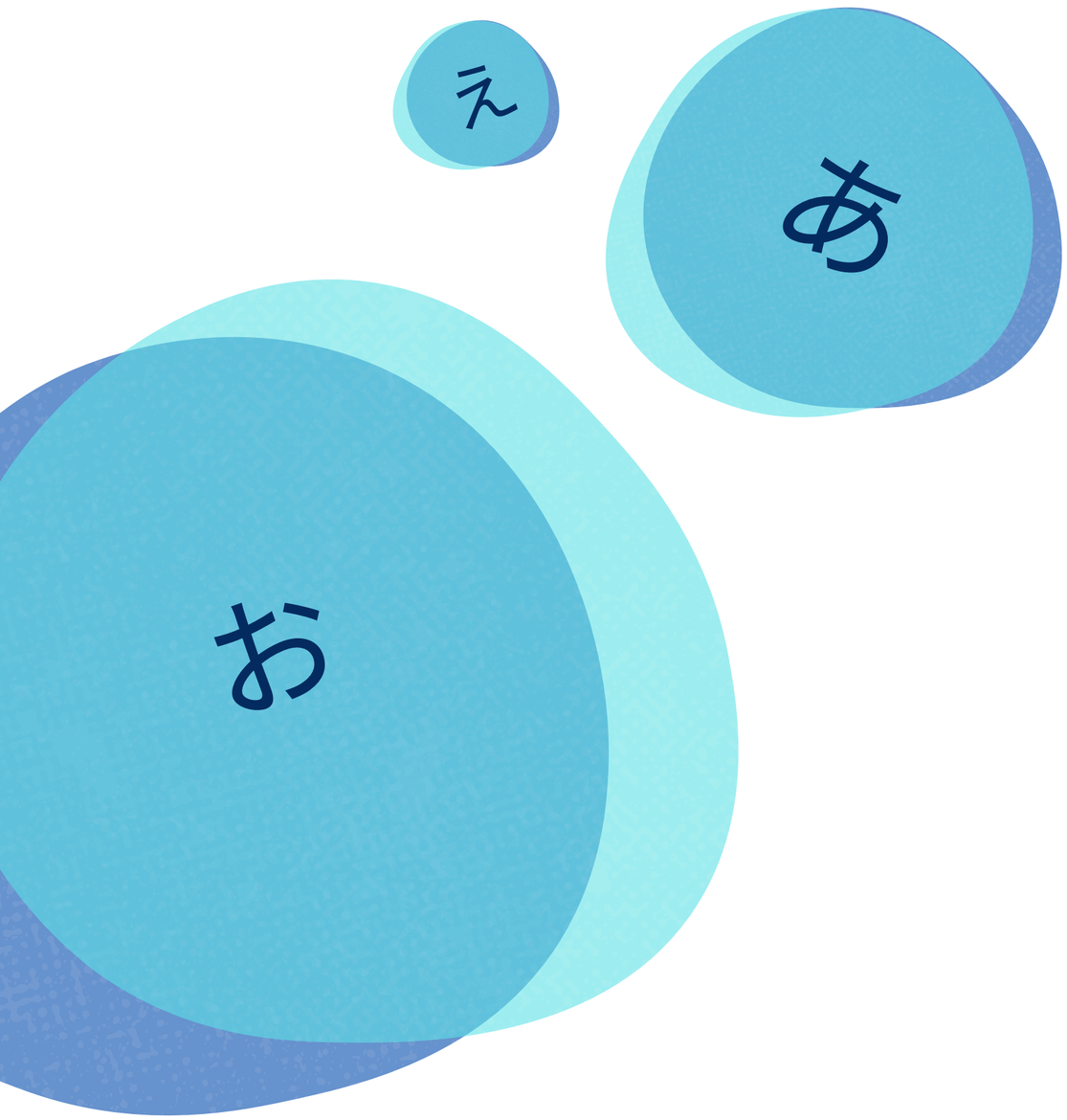
USANDO PYTHON Y NLP PARA APRENDER LENGUAJES

A HABLAR

- Historia
- NLP y NLU
- Spacy y Stanza
- Proyecto



HISTORIA

Como un chiste llega demasiado lejos

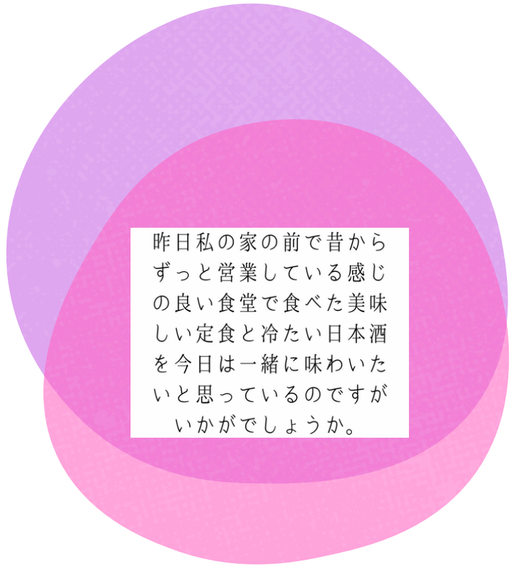
POR QUE JAPONES

Simple, sonaba cool. (mala decisión)

Empezar con duolingo

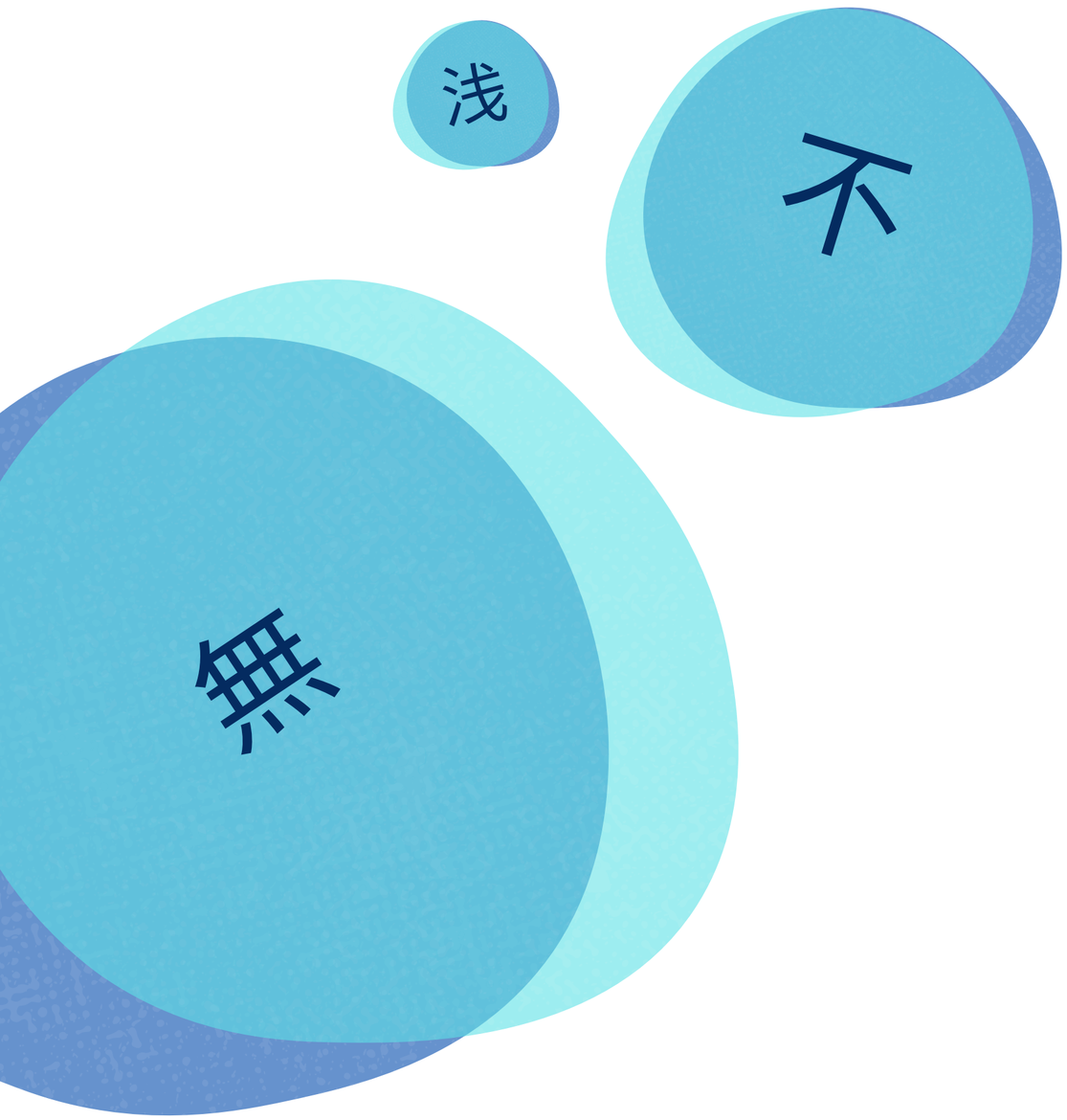

Japanese: (私は) 先月日本の本社で佐藤さんが出席した会議について、何の報告も受けていないです。
English: I haven't heard anything about the meeting Sato-san attended last month at the HQ in Japan.

Irse a Japón a estudiar



Sufrir con los sistemas de escritura

Pasar horas estudiando para aun asi sufrir leyendo



NLP Y NLU

Los fundamentos.

NLP

El campo de la AI dedicado a procesar, generar y analizar lenguaje humano.

NLU

La rama de NLP que se enfoca en entender el lenguaje natural, no solo procesar.

UPOS

Etiquetas universales para las categorías gramaticales de las palabras

LEMMA

La forma base de una palabra.

TAG

Elementos del habla especificos del lenguaje

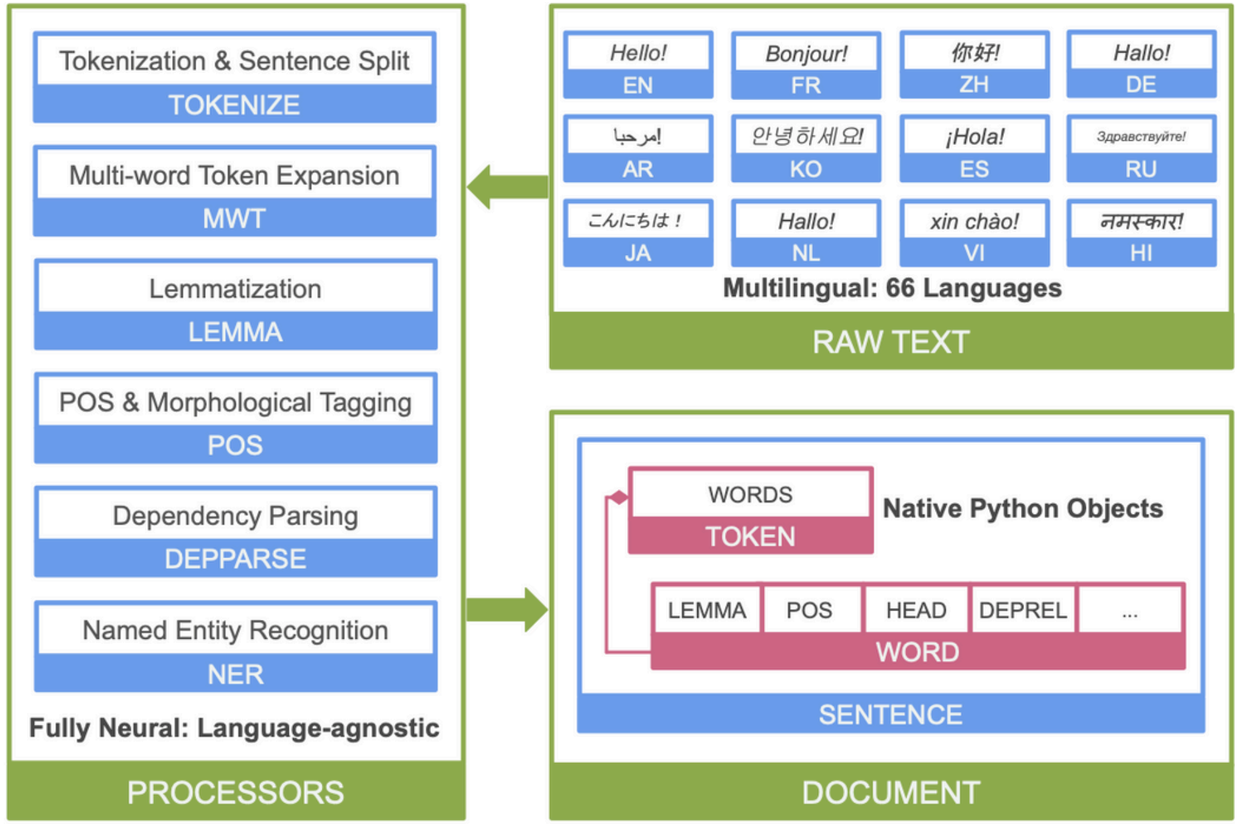
DEPENDENCIA

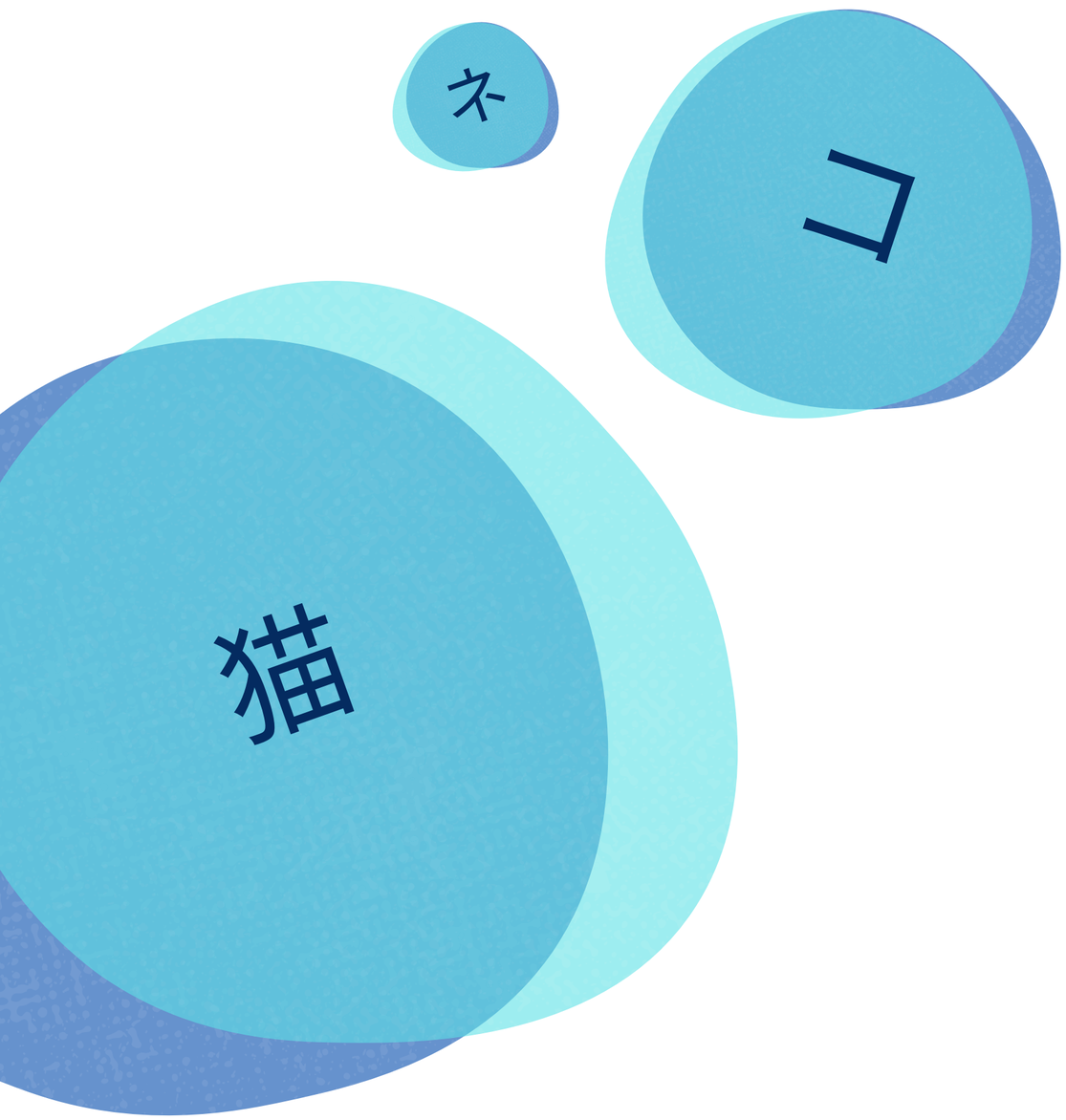
Relacion entre los diferentes tokens, es decir la cabeza del token

ENT_TYPE

Tipo de entidades, como nombres de ciudades, etc.

Una Pipeline tipica de NLP





SPACY Y STANZA

La magia en dias aburridos

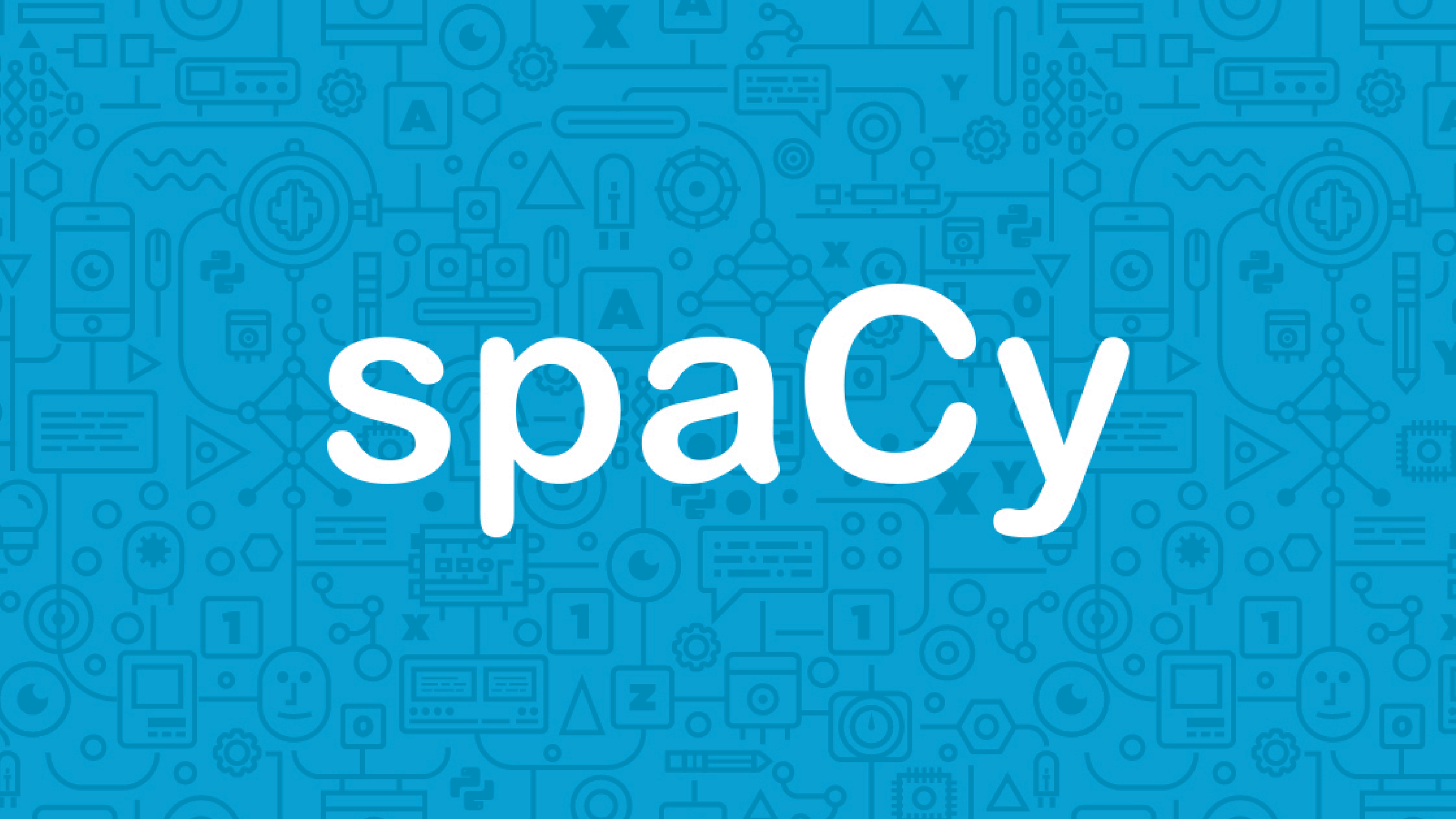
Stanza

¿QUE ES?

Un modulo de python para analisis del lenguaje natural.

DEMO

stanza


spaCy

POR QUE

Libreria para NLP avanzado que se puede usar en producción.

QUE ESTA COOL

Con un codigo bastante simple se puede aplicar todo lo que hemos visto.

FEATURES

- Amplio soporte de lenguajes
- Pipelines de NLP pre-entrendas
- Buena velocidad
- Soporte para modelos propios en TF y PyTorch

VENTAJAS

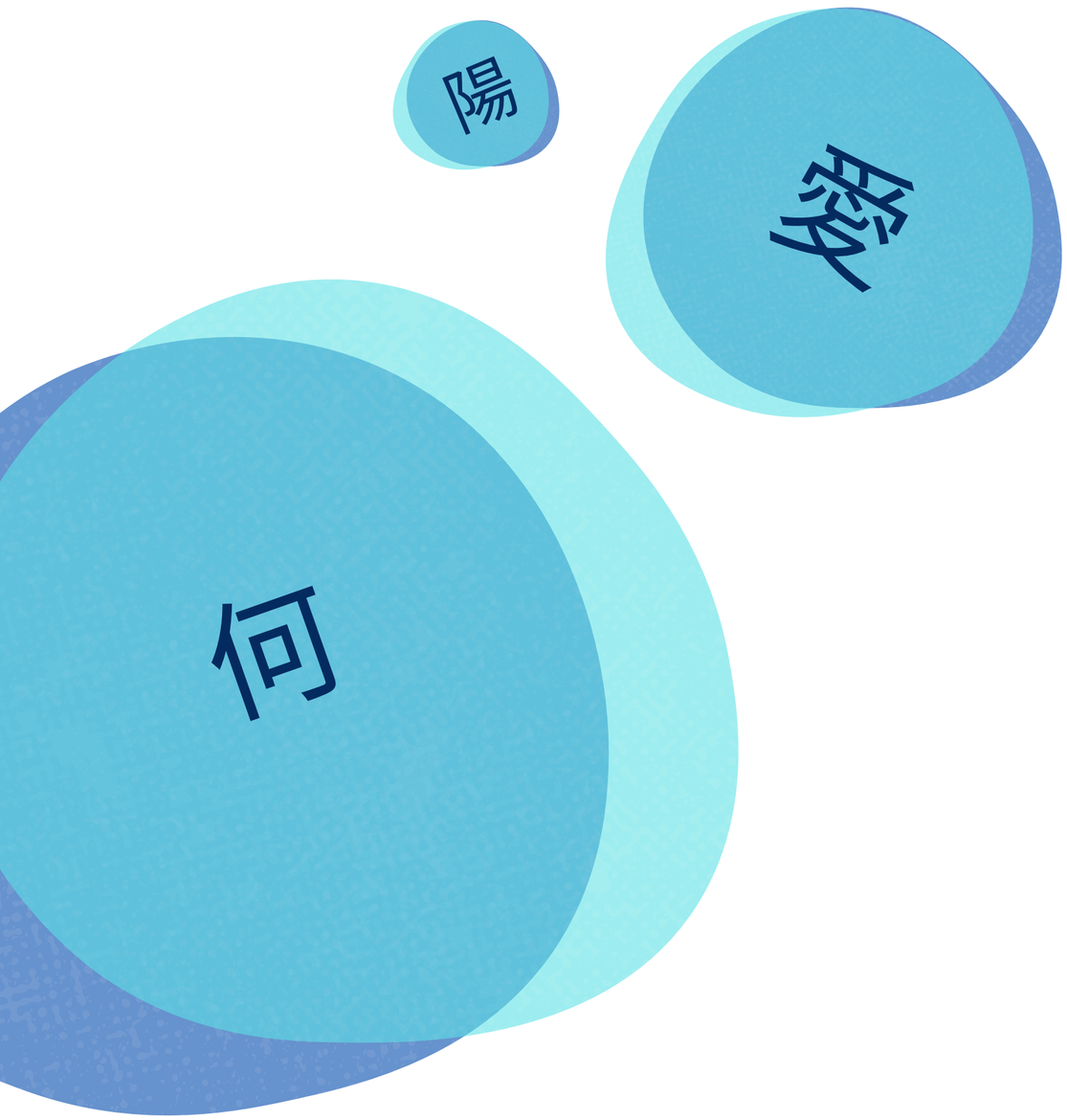
- Facil de usar como principiante
- Bueno para construir una aplicación entera.
- Eficiente en GPU y CPU
- Probar diferentes NN facilmente

DESVENTAJAS

- No tiene funciones generativas
- No esta orientada a hacer research

CODE SAMPLE

```python
class NLPService:  2 usages  Miguel Necoechea +1 *
    def __init__(self,spacy_model: str = "ja_ginza_bert_large"):  Miguel Necoechea *
        spacy.prefer_gpu()
        self.nlp = spacy.load(spacy_model)
        self.fallback = kakasi()
        self.japanese_detector = JapaneseDetector()

    def _convert_reading(self, surface: str) -> str:  Miguel Necoechea
        fb = self.fallback.convert(surface)
        return ''.join(item["kana"] for item in fb) if fb else None

    def _token_to_dict(self, token) -> Dict:  1 usage  Miguel Necoechea *
        surface = token.text
        reading = token.morph.get("Reading")

        if reading:
            reading = reading[0]

        is_japanese = self.japanese_detector.is_japanese(surface)

        return {
            "surface":  surface,
            "reading":  reading,
            "lemma":    token.lemma_,
            "pos":      token.pos_,
            "tag":      token.tag_,
            "dep":      token.dep_,
            "head":     token.head.text,
            "morph":    token.morph.to_dict(),
            "offset":   token.idx,
            "ent_iob":  token.ent_iob_,
            "ent_type": token.ent_type_,
            "is_japanese": is_japanese
        }

    def tokenize_batch(self, texts: List[str]) -> List[List[Dict]]:  1 usage  Miguel Necoechea +1
        docs = self.nlp.pipe(texts, batch_size=max(1, len(texts)))
        return [
            [self._token_to_dict(tok) for tok in doc]
            for doc in docs
        ]
```



PROYECTO

¿Bueno y todo esto pa' que?

Pensemos en Syntax Highlighting

```python
class NLPService:
    def __init__(self,spacy_model: str = "ja_ginza_bert_large"):
        spacy.prefer_gpu()
        self.nlp = spacy.load(spacy_model)
        self.fallback = kakasi()
        self.japanese_detector = JapaneseDetector()

    def _convert_reading(self, surface: str) -> str:
        fb = self.fallback.convert(surface)
        return ''.join(item["kana"] for item in fb) if fb else None

    def _token_to_dict(self, token) -> Dict:
        surface = token.text
        reading = token.morph.get("Reading")

        if reading:
            reading = reading[0]

        is_japanese = self.japanese_detector.is_japanese(surface)

        return {
            "surface":  surface,
            "reading":  reading,
            "lemma":    token.lemma_,
            "pos":      token.pos_,
            "tag":      token.tag_,
            "dep":      token.dep_,
            "head":     token.head.text,
            "morph":    token.morph.to_dict(),
            "offset":   token.idx,
            "ent_iob":  token.ent_iob_,
            "ent_type": token.ent_type_,
            "is_japanese": is_japanese
        }

    def tokenize_batch(self, texts: List[str]) -> List[List[Dict]]:
        docs = self.nlp.pipe(texts, batch_size=max(1, len(texts)))
        return [
            [self._token_to_dict(tok) for tok in doc]
            for doc in docs
        ]
```

El color lo hace mas fácil (casi siempre)

Y asi nacio Complexa

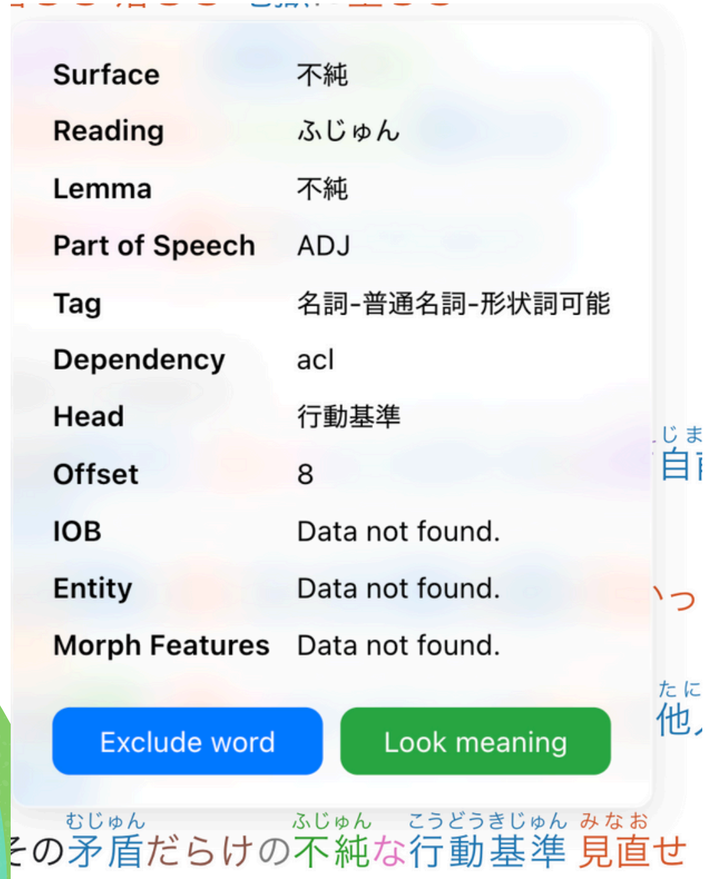
¿QUE ES?

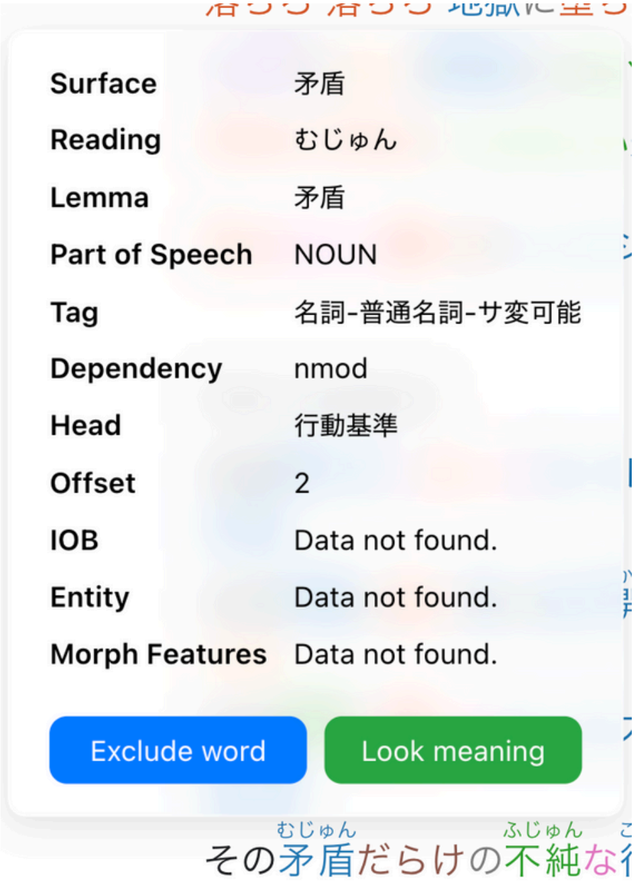
Complexa es una extension de Chrome que hace un análisis morfológico del texto en japones de paginas web para ayudar en el aprendizaje del japonés.

その分厚く肥ったルールブックはお前自前の妄想
その恩着せがましいお説教は鏡に向かってどうぞ
その無様に狂ったルールブックをすぐに他人に強要
その矛盾だらけの不純な行動基準 手放せ

その分厚く肥ったルールブックはお前自前の妄想
その恩着せがましいお説教は鏡に向かってどうぞ
その無様に狂ったルールブックをすぐに他人に強要
その矛盾だらけの不純な行動基準 手放せ

Y la información de cada token


Surface 不純
Reading ふじゅん
Lemma 不純
Part of Speech ADJ
Tag 名詞-普通名詞-形状詞可能
Dependency acl
Head 行動基準
Offset 8
IOB Data not found.
Entity Data not found.
Morph Features Data not found.
Exclude word
Look meaning
その矛盾だらけの不純な行動基準 見直せ


Surface 矛盾
Reading むじゅん
Lemma 矛盾
Part of Speech NOUN
Tag 名詞-普通名詞-サ変可能
Dependency nmod
Head 行動基準
Offset 2
IOB Data not found.
Entity Data not found.
Morph Features Data not found.
Exclude word
Look meaning
その矛盾だらけの不純な

Live Demo!

私のプレゼンテーションを最後までお聞きいただき、誠にありがとうございました

O muchas gracias por escuchar pues

Q&A