NLP

Word Embeddings

Docentes:
Dr. Rodrigo Cardenas Szigety
Dr. Nicolás Vattuone

emails: rodrigo.cardenas.sz@gmail.com
nicolas.vattuone@gmail.com

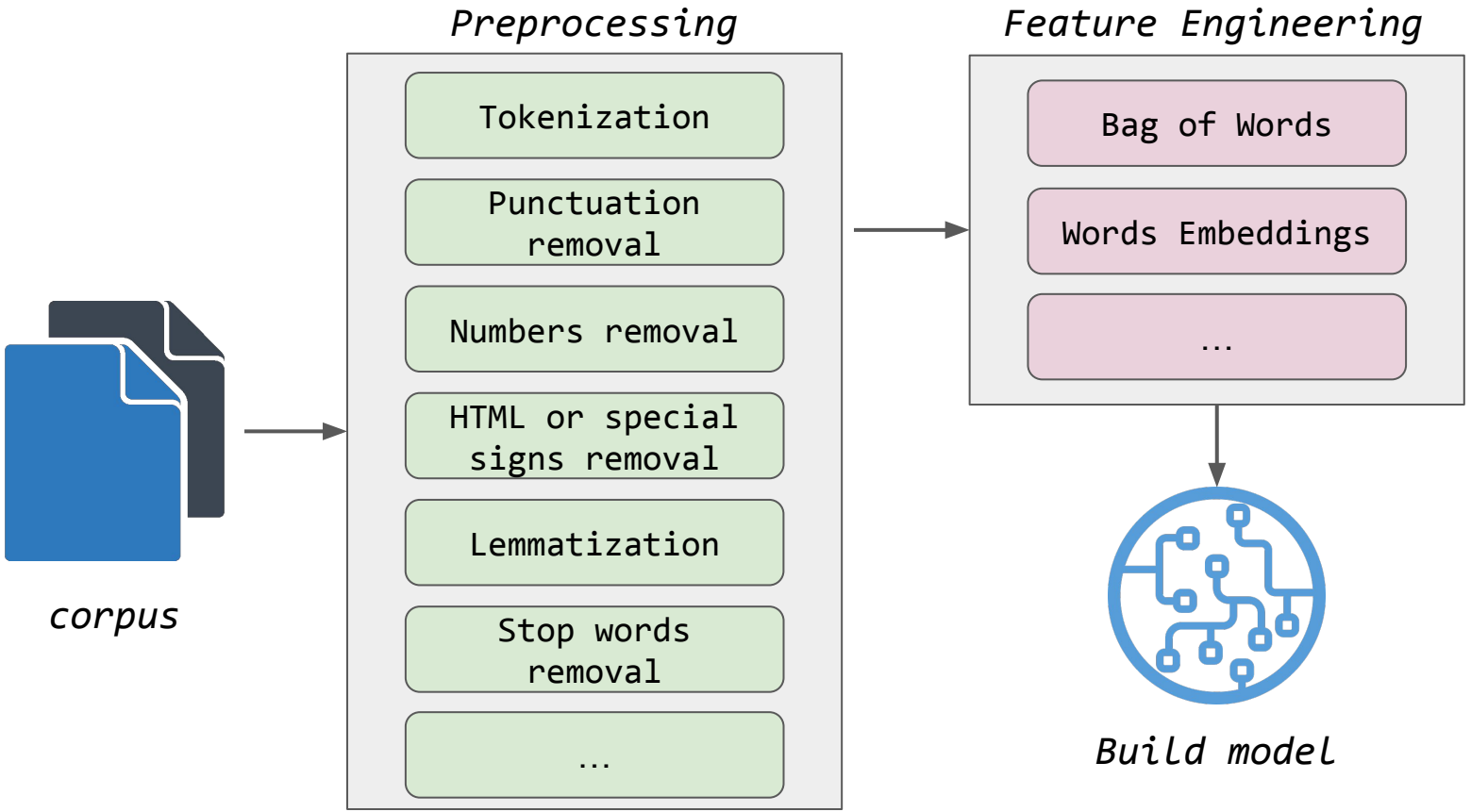
Programa de la materia

Clase 1: Introducción a NLP, Vectorización de documentos.
Clase 2: Pre-procesamiento de texto. Word embeddings.
Clase 3: Modelos no-BOW I: convolucionales y recurrentes. Generación de secuencias.
Clase 4: Modelos no-BOW II: mecanismo de atención.
Clase 5: Modelos Seq2seq.
Clase 6: Transformers.
Clase 7: Grandes modelos de lenguaje. RAG.
Clase 8: Otros temas: Image captioning. ASR y TTS.

*Unidades con desafíos a presentar al finalizar el curso.
*Último desafío y cierre del contenido práctico del curso.

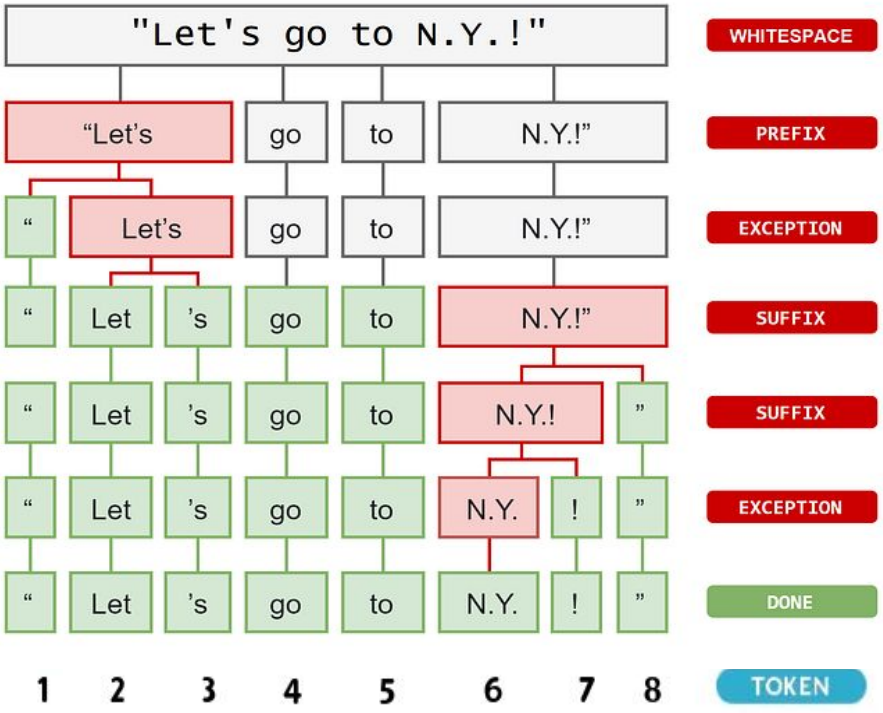
Preprocesamiento de texto

LINK GLOSARIO



Segmentar y tokenizar

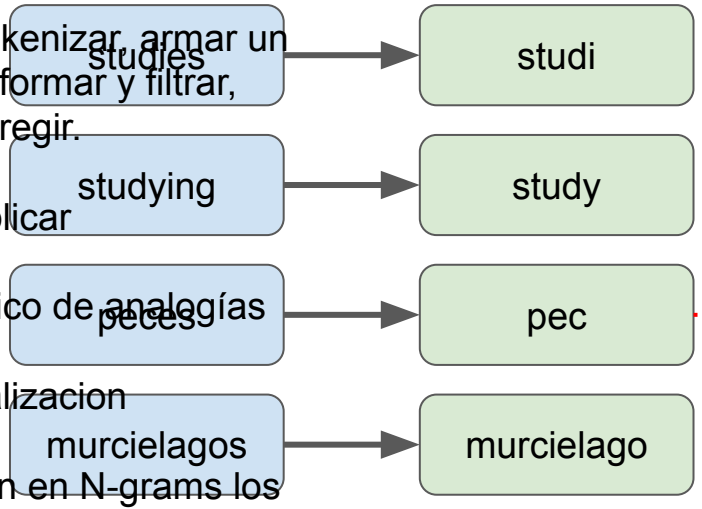
Proceso en el cual una oración o documento es segmentado en términos individuales. Una vez finalizada la segmentación cada término único es referenciado mediante un token.



Derivado (steeming)

Aplica reglas de eliminación de patrones recurrentes de la lengua. El resultado es una palabra truncada que no será necesariamente la raíz morfológica de la palabra.

Regla: Eliminar los sufijos



plural irregular

especimen, especímenes
régimen, regímenes

segmentar y tokenizar, armar un ejemplo. Transformar y filtrar, normalizar, corregir.

n- gramas, explicar

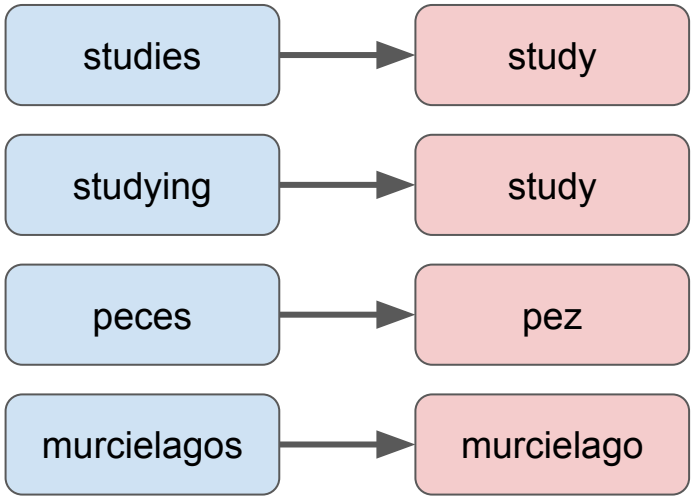
corregir el grafico de analogías

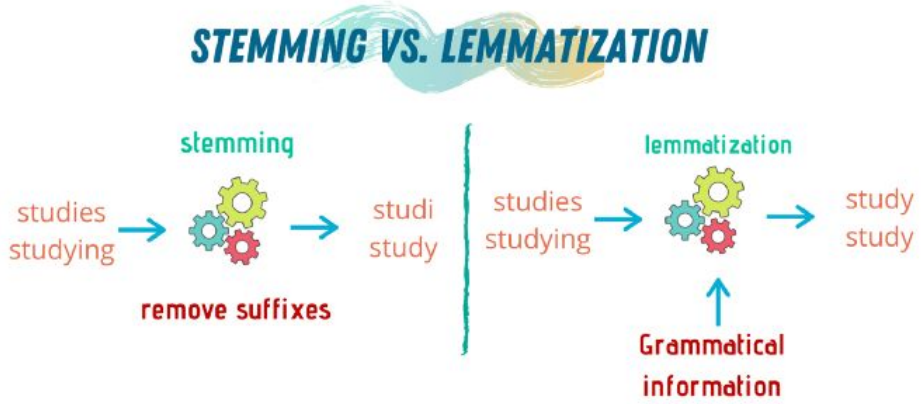
Agregar normalizacion

Cómo se toman en N-grams los ejemplos positivos y negativos

Lematización (lemmatization)

Devuelve la raíz morfológica de una palabra. Para ello se necesita un diccionario del idioma con todas las declinaciones posibles de las palabras raíz.



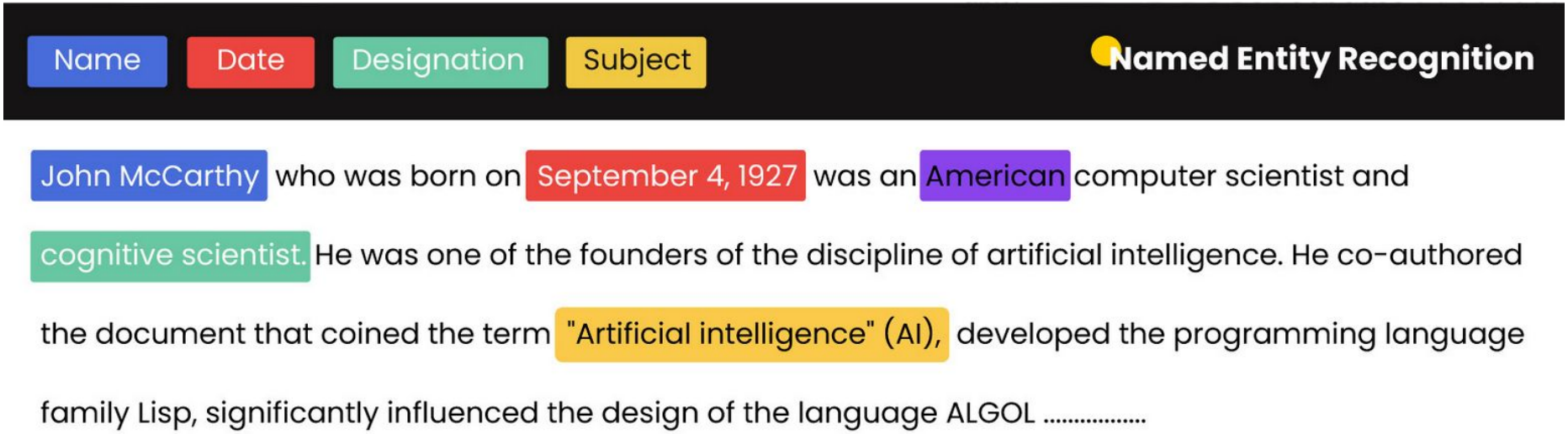


Parts-of-speech (POS) tagging

POS es el proceso de clasificar cada término en un texto en sus categorías gramaticales, etiquetándolos por ejemplo como ***sustantivo (noun), verbo (verb), adjetivo (adj),*** *etc*

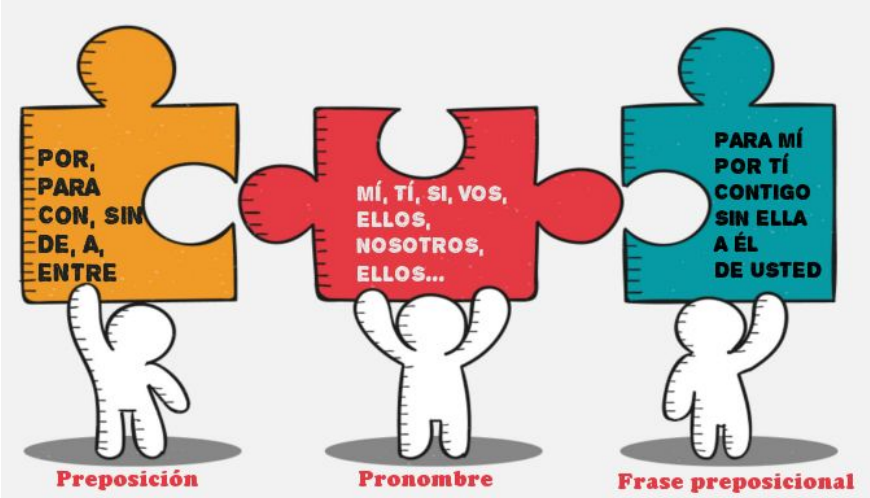
Named-entity recognition (NER)

NER es el proceso de clasificar nombres propios de entidades en categorías predefinidas a las cuales pertenecen.



Stop words

Palabras que no aportan valor al significado de una oración ya que son muy frecuentes o comunes en el lenguaje



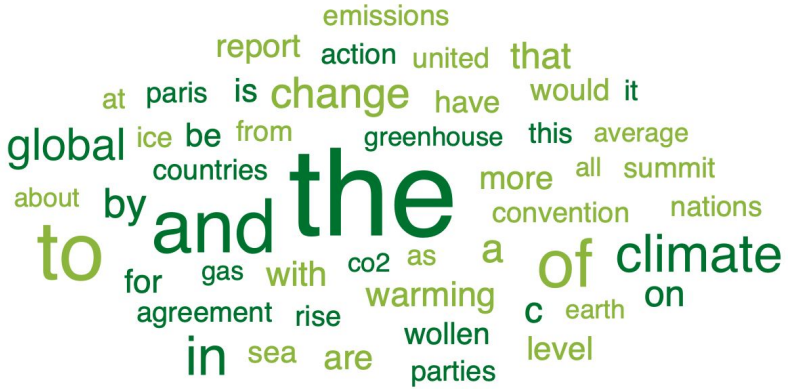
*Argentina, oficialmente, República Argentina, es **un** país soberano **de** América **del** Sur, ubicado **en el** extremo sur **y** sudeste **de** dicho subcontinente. Adopta **la** forma **de** gobierno republicana, democrática, representativa **y** federal.*

Argentina, oficialmente, República Argentina, es país soberano América Sur, ubicado extremo sur sudeste dicho subcontinente. Adopta forma gobierno republicana, democrática, representativa federal.

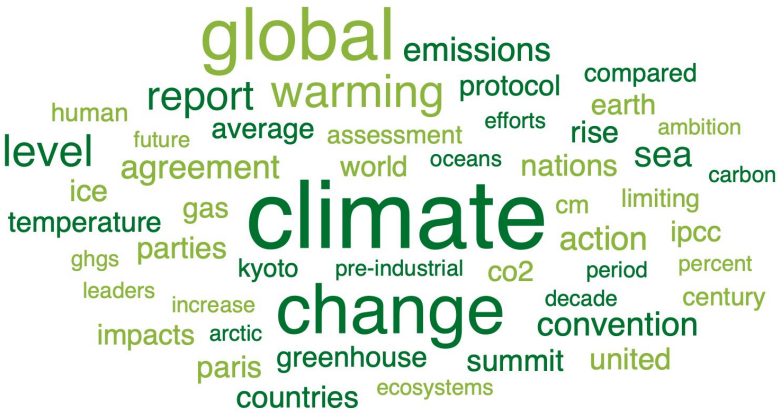
Stop words

Analizar un texto relacionado con calentamiento global (global climate)

Texto con Stop Words

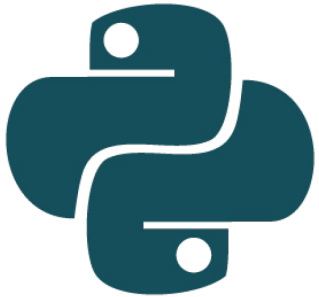


Texto sin Stop Words



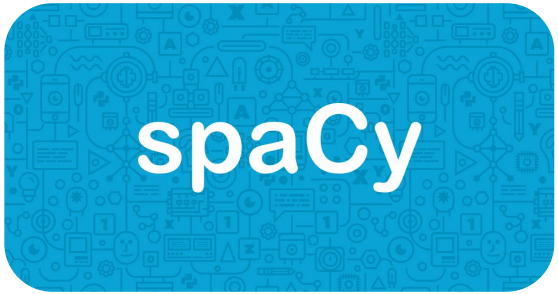
Las Stop Words pueden depender del contexto del corpus.

Librerías de NLP



NLTK

- Gran comunidad de desarrollo
- No soporta GPU
- Más optimizada en CPU
- Más lenta en gran volúmenes de datos o operaciones



- Más moderna e implementa los últimos features
- Soporta GPU
- Menos optimizada en CPU
- Más rápida en gran volúmenes de datos o operaciones (GPU)

NLTK

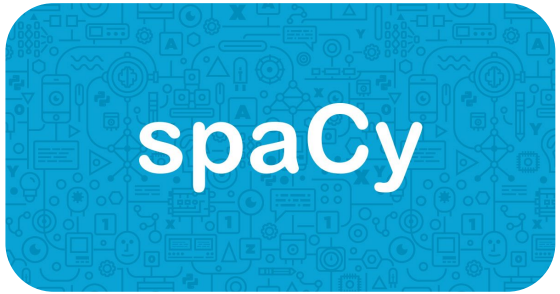
LINK PÁGINA

"Librería por excelencia de procesamiento de lenguaje natural para Python"
Inicios 2009

Implementa una tool/algoritmo para cada etapa de preprocesamiento de NLP

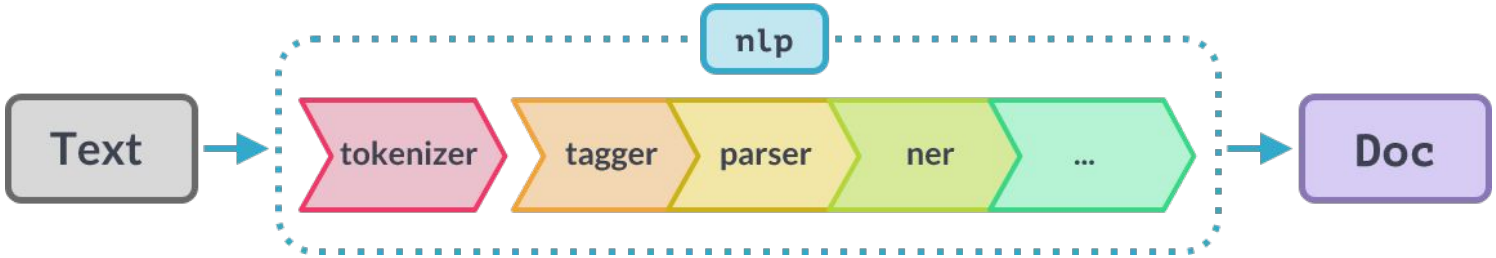
spaCy

LINK PÁGINA



Inicios 2015

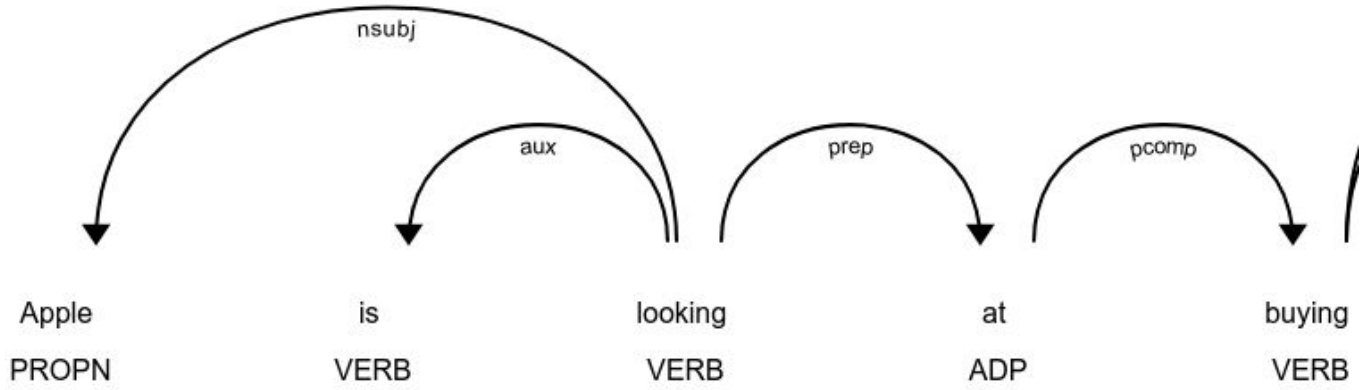
- Support for **72+ languages**
- **80 trained pipelines** for 24 languages
- Multi-task learning with pretrained **transformers** like BERT
- Pretrained **word vectors**
- State-of-the-art speed
- Production-ready **training system**



spaCy

LINK PÁGINA

```
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")

for token in doc:
    print(token.text, token.lemma_, token.pos_, token.tag_, token.dep_,
            token.shape_, token.is_alpha, token.is_stop)
```



Un resumen de todo lo visto

POS TAG DEP

TEXT	LEMMA	POS	TAG	DEP	SHAPE	ALPHA	STOP
Apple	apple	PROPN	NNP	nsubj	Xxxxx	True	False
is	be	AUX	VBZ	aux	xx	True	True
looking	look	VERB	VBG	ROOT	xxxx	True	False
at	at	ADP	IN	prep	xx	True	True
buying	buy	VERB	VBG	pcomp	xxxx	True	False
U.K.	u.k.	PROPN	NNP	compound	X.X.	False	False
startup	startup	NOUN	NN	dobj	xxxx	True	False
for	for	ADP	IN	prep	xxx	True	True
$	$	SYM	$	quantmod	$	False	False
1	1	NUM	CD	compound	d	False	False
billion	billion	NUM	CD	pobj	xxxx	True	False

Preprocesamiento

Link al Colab

LINK

Problemas con CountVectorizer/OHE/TF-IDF

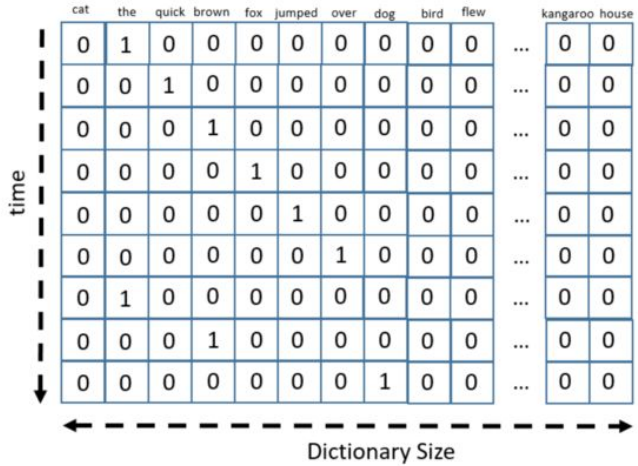
Textos de significado similar pueden ser “ortogonales”

Estoy viajando en colectivo

Voy arriba del bus

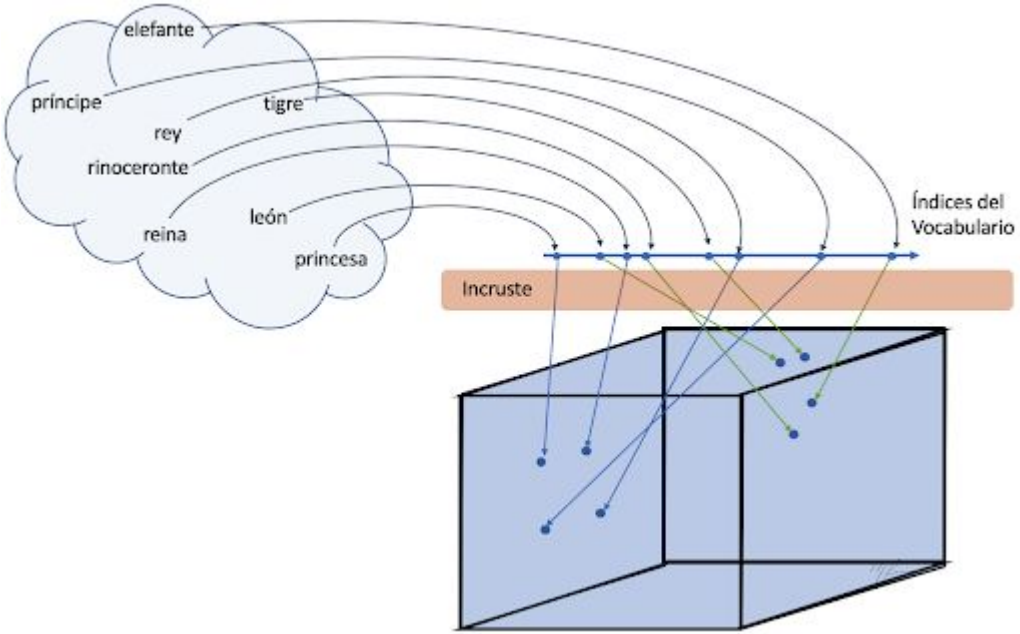
La dimensión de los vectores depende del tamaño del vocabulario

No aprovechamos la dimensionalidad



Embeddings

Un embedding es una representación densa de palabras en un espacio vectorial continuo.

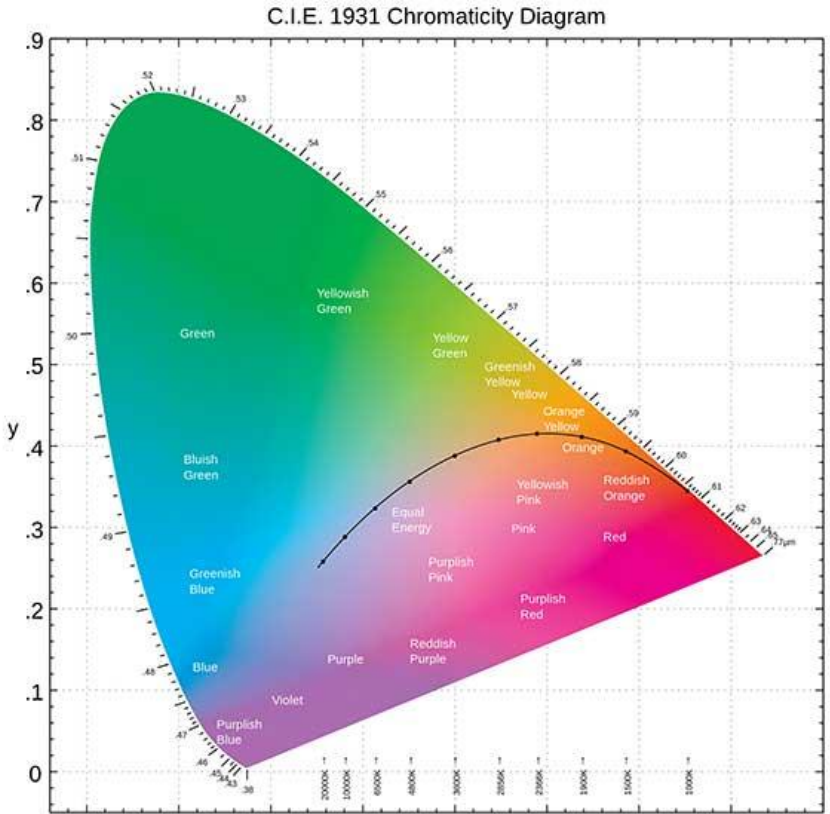


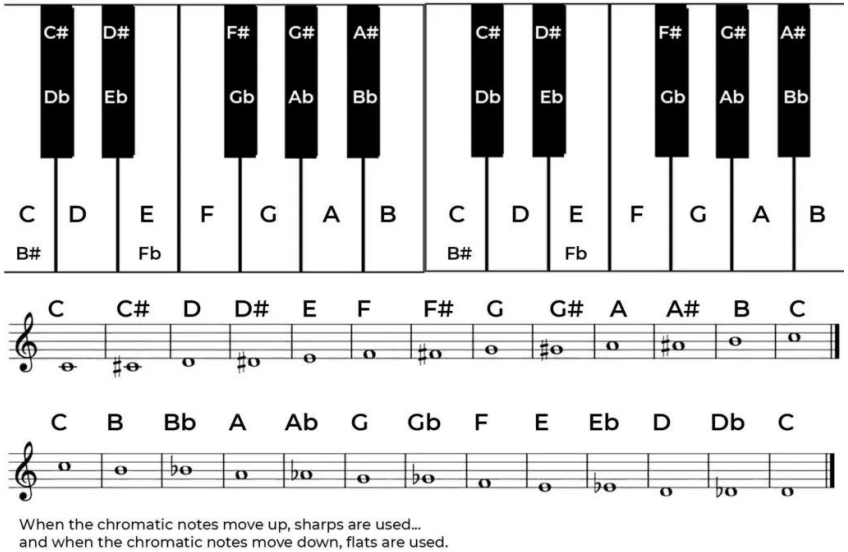
Propiedades buscadas:

1. ***Representación compacta***
2. ***Similaridad semántica***
3. ***Comprensión de contexto***

Representaciones densas intuitivas

La relación entre conjuntos de palabras y representaciones densas es más clara para palabas que designan experiencias sensoriales:



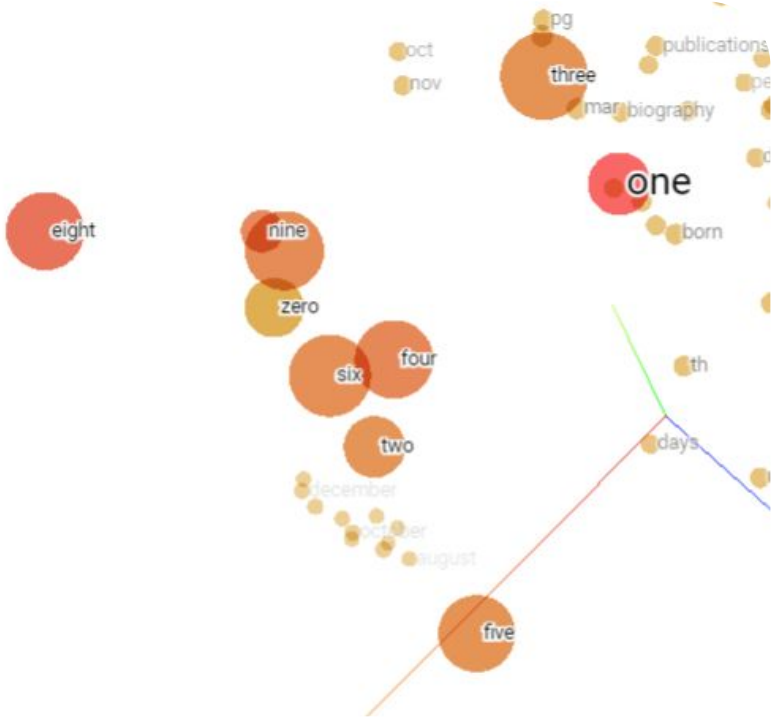


When the chromatic notes move up, sharps are used...
and when the chromatic notes move down, flats are used.

Word Embeddings

Las palabras que tienen un significado similar tendrán una representación similar como embeddings

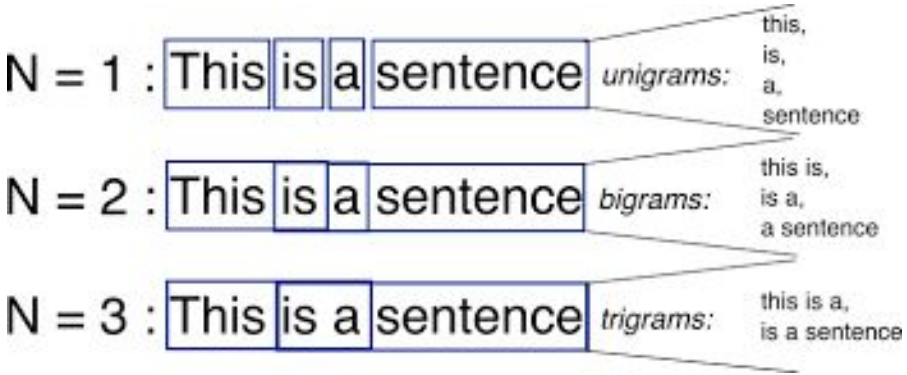
http://projector.tensorflow.org/



N-GRAM

"Subsecuencia de N elementos de una secuencia dada"

Otra forma de agrupar las palabras distinto a word2vec (N=1, unigram) en donde se busca aumentar el poder de generalización y a su vez poder hacer más variado el vocabulario.



Character-level unigrams

Text	Token Sequence	Token Value
Dogs	1	D
Dogs	2	o
Dogs	3	g
Dogs	4	s

Character-level bigrams

Text	Token Sequence	Token Value
Dogs	1	Do
Dogs	2	og
Dogs	3	gs

Character-level trigrams

Text	Token Sequence	Token Value
Dogs	1	Dog
Dogs	2	ogs

GloVe y fastText

Embeddings pre-entrenados basados en diferentes enfoques:

GloVe

Tokenización basada en palabras

Entrenado con textos de Wikipedia, Common Crawl y GigaWord 5

Se basa en calcular la matriz de co-ocurrencia de palabras y estimar el cociente de probabilidad de aparición.

fastText

Tokenización basada en N-Grams de caracteres (3 a 6). Mejora la interpretación de sufijos y prefijos

Entrenado con una colección de 8 corpus (portales de noticias, reviews, WIkipedia)

Basado en word2vec (CBOW/Skip-Gram)

Puede crear un embedding de una palabra que nunca vió

Matriz de co-ocurrencia

Obtención del corpus: Reuní las letras de sus canciones desde fuentes en línea, como letras.top y Buenamusica.com.

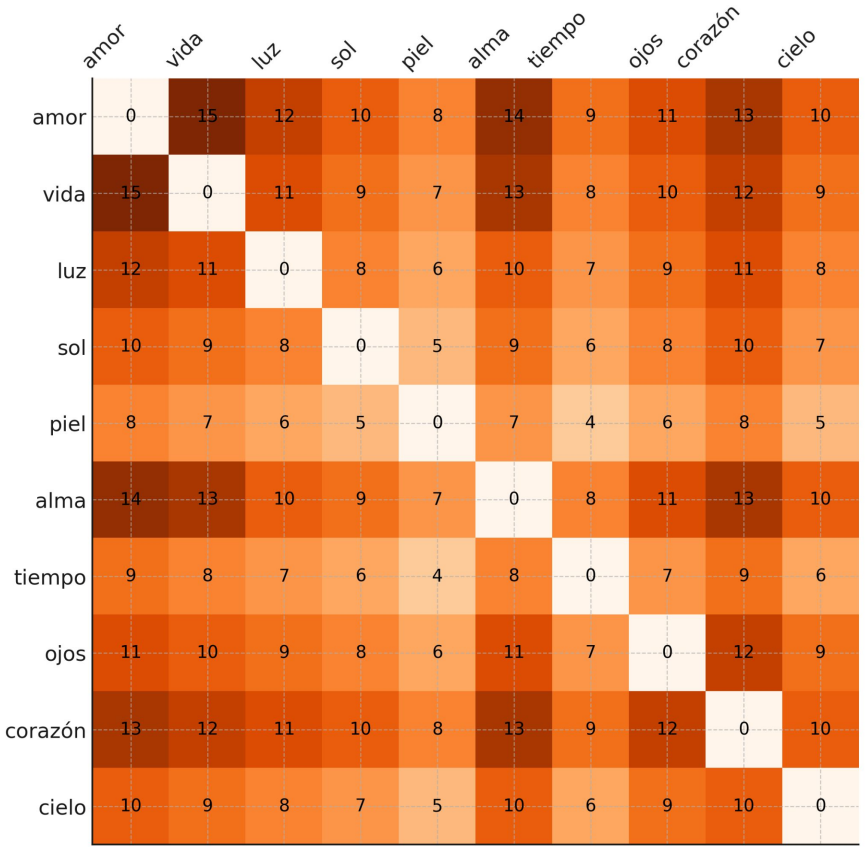
Preprocesamiento del texto:

- Convertí todo el texto a minúsculas.
- Eliminé signos de puntuación.
- Tokenicé el texto para dividirlo en palabras individuales.
- Eliminé palabras vacías comunes en español, como "el", "la", "y", etc.

Cálculo de frecuencias: Conté la frecuencia de cada palabra y seleccioné las 10 más comunes.

Construcción de la matriz de co-ocurrencia:

- Definí una ventana de contexto de 5 palabras alrededor de cada término.
- Conté cuántas veces cada par de palabras apareció dentro de la misma ventana.



Matriz de co-ocurrencia

Obtención del corpus: Reuní las letras de sus canciones desde fuentes en línea, como letras.top y Buenamusica.com.

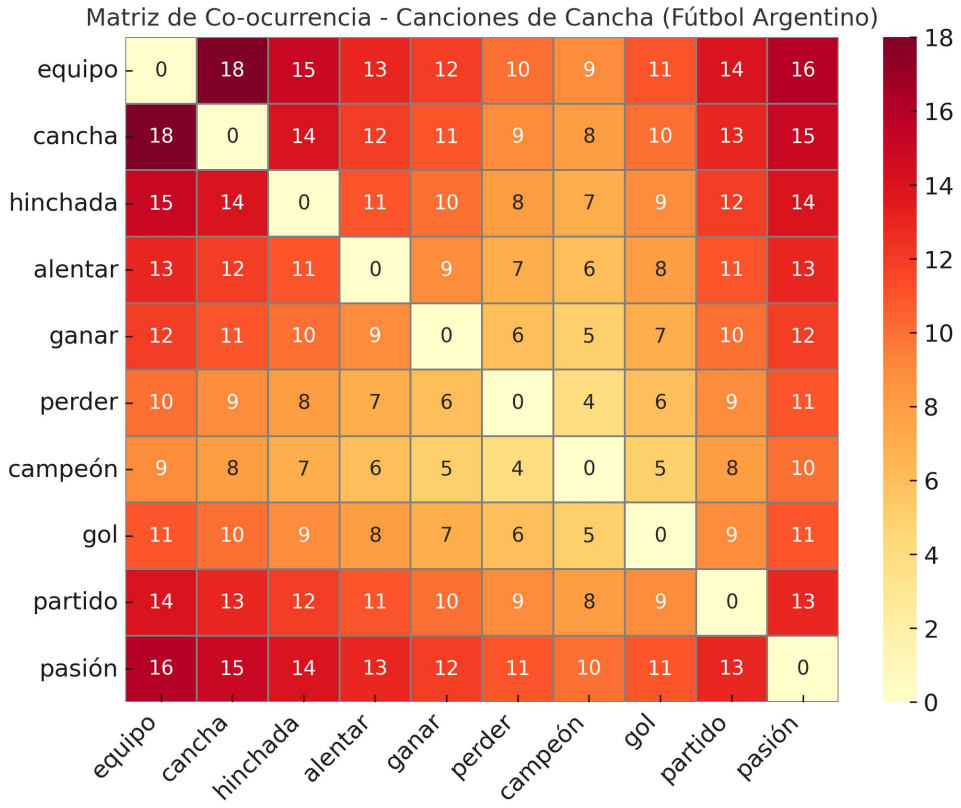
Preprocesamiento del texto:

- Convertí todo el texto a minúsculas.
- Eliminé signos de puntuación.
- Tokenicé el texto para dividirlo en palabras individuales.
- Eliminé palabras vacías comunes en español, como "el", "la", "y", etc.

Cálculo de frecuencias: Conté la frecuencia de cada palabra y seleccioné las 10 más comunes.

Construcción de la matriz de co-ocurrencia:

- Definí una ventana de contexto de 5 palabras alrededor de cada término.
- Conté cuántas veces cada par de palabras apareció dentro de la misma ventana.



GloVe y fastText

Embeddings pre-entrenados basados en diferentes enfoques:

GloVe

Tokenización basada en palabras

Entrenado con textos de Wikipedia, Common Crawl y GigaWord 5

Se basa en calcular la matriz de co-ocurrencia de palabras y estimar el cociente de probabilidad de aparición.

fastText

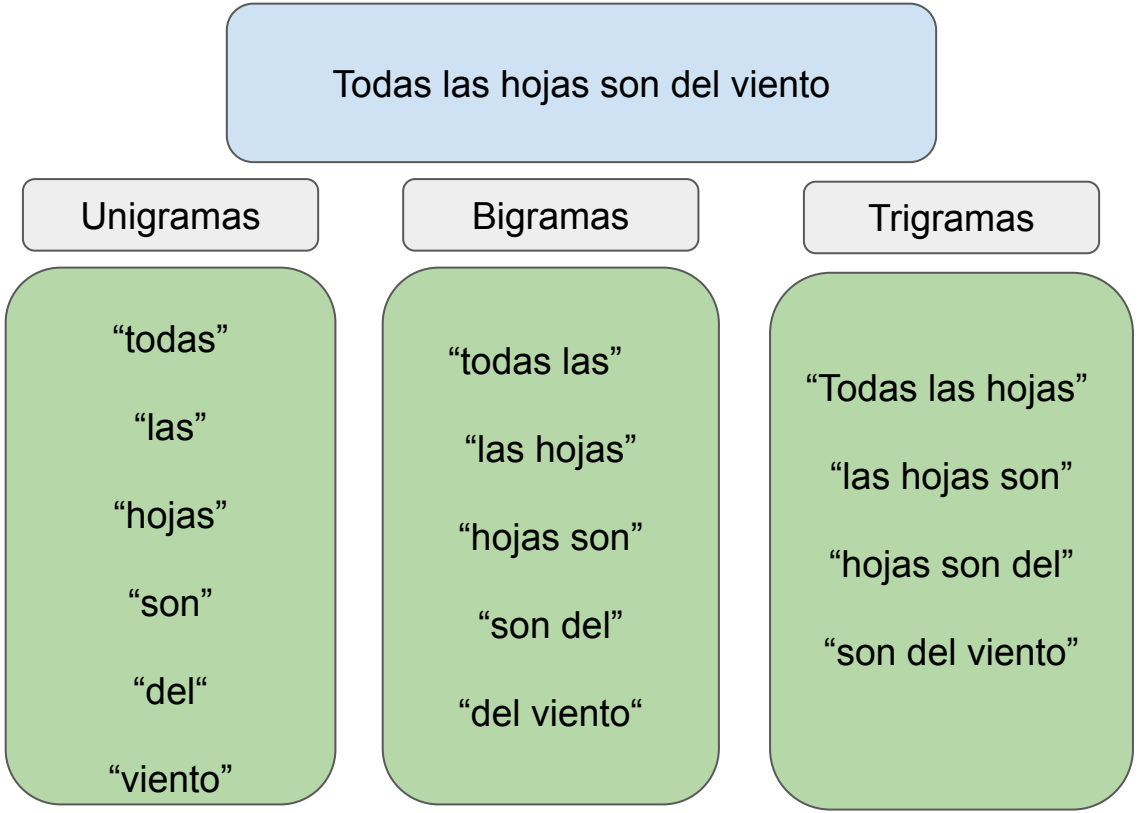
Tokenización basada en N-Grams de caracteres (3 a 6). Mejora la interpretación de sufijos y prefijos

Entrenado con una colección de 8 corpus (portales de noticias, reviews, WIkipedia)

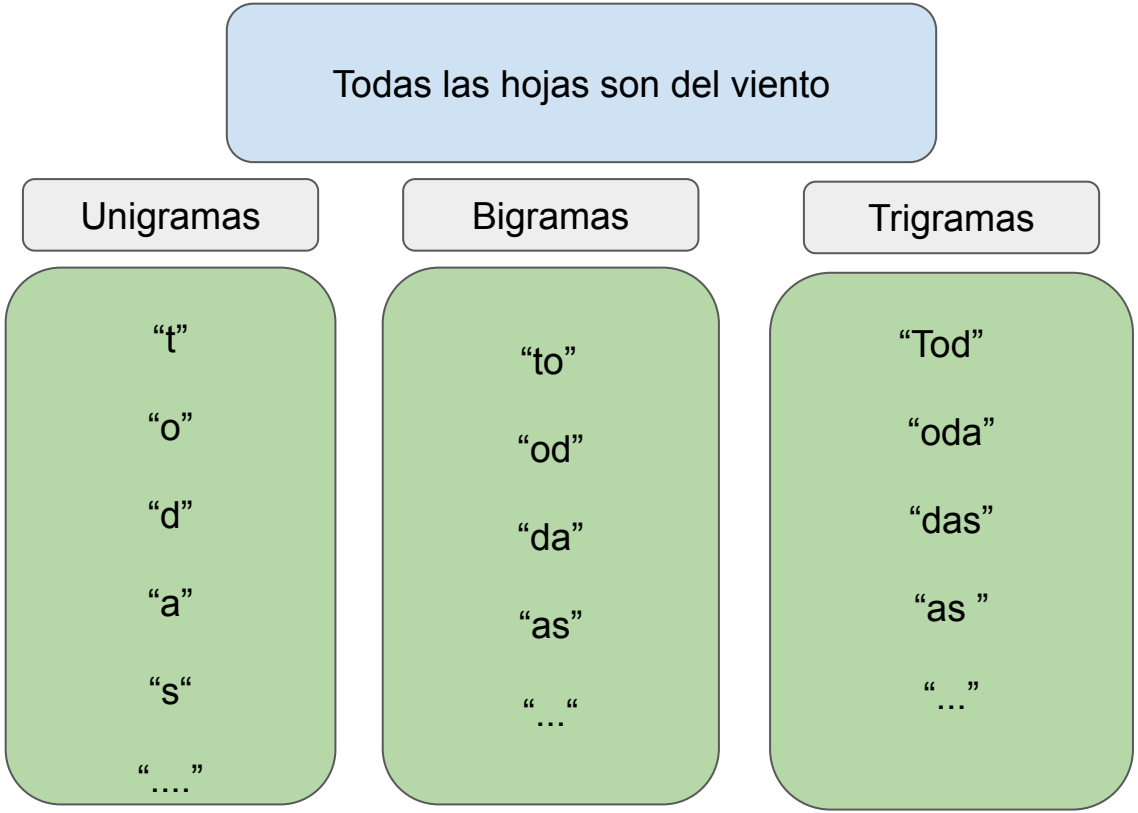
Basado en word2vec (CBOW/Skip-Gram)

Puede crear un embedding de una palabra que nunca vió

N-gramas (por palabras)



N-gramas (por caracteres)



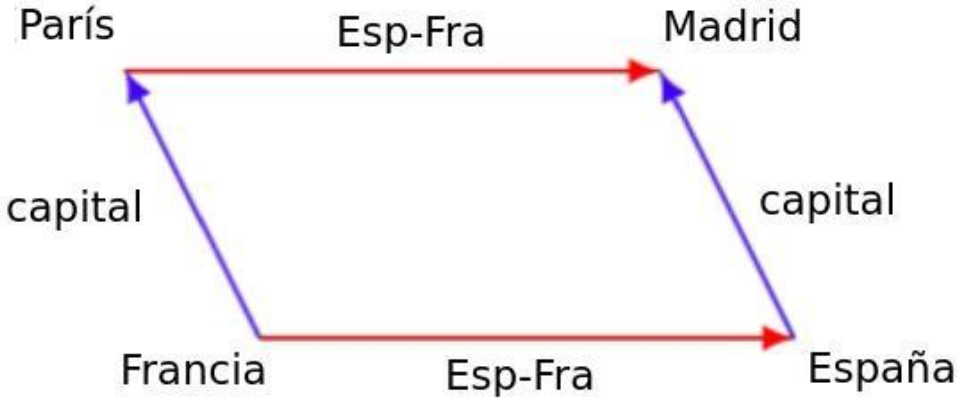
Operaciones con Embeddings: tests de analogías

Una forma de testear la calidad de embeddings es probar su desempeño en tests de analogías:

París es a Francia lo que Madrid es a España.
Madrid y París corresponden a España y Francia

$$\overrightarrow{Paris} - \overrightarrow{Francia} \approx \overrightarrow{Madrid} - \overrightarrow{Espana}$$

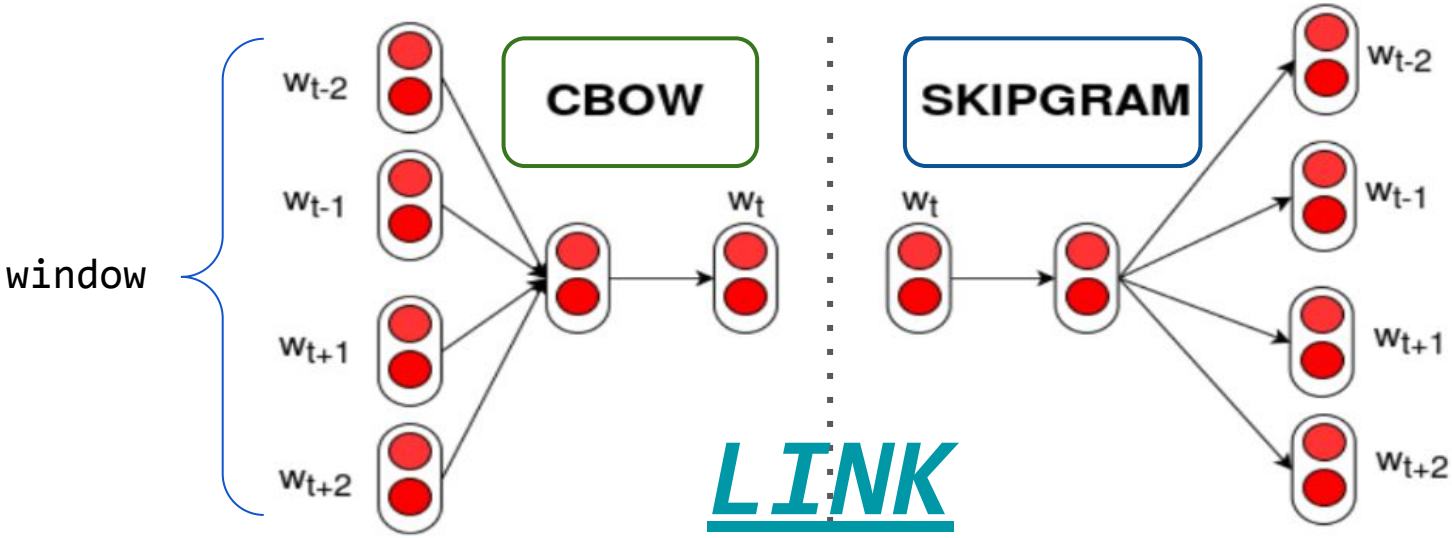
$$simcos(\overrightarrow{Paris} - \overrightarrow{Francia}, \overrightarrow{Madrid} - \overrightarrow{Espana}) \approx 1$$



Embeddings Glove y Fasttext

¿Cómo podemos crear nuestros word Embeddings?

Aprendiendo (con redes neuronales) vectores para cada palabra que maximicen la relación entre las palabras de contexto y la palabra objetivo. Esto es lo que se implementó en la librería ***word2vec****.*



LINK

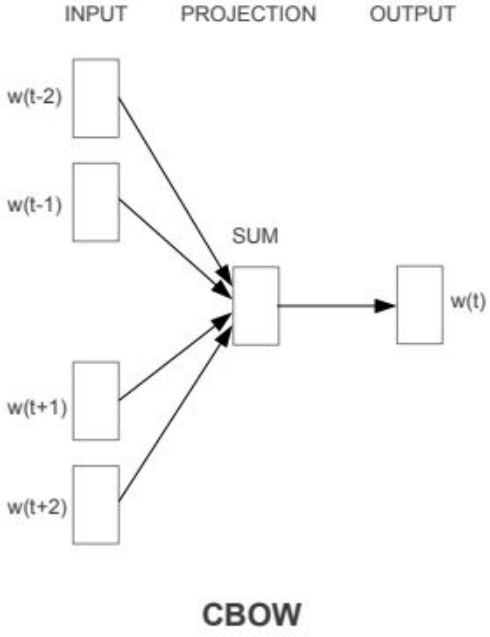
$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_t|w_{t+j})$$

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j}|w_t)$$

Se utiliza ***OneHotEncoding*** como representación del texto a la entrada del modelo

Continuous Bag of Words Model (CBOW)

Utiliza como entrada el contexto de la palabra objetivo (palabras a izquierda y derecha de ella). El tamaño de la ventana determina cuántas palabras se tomarán para contextualizar el embedding.

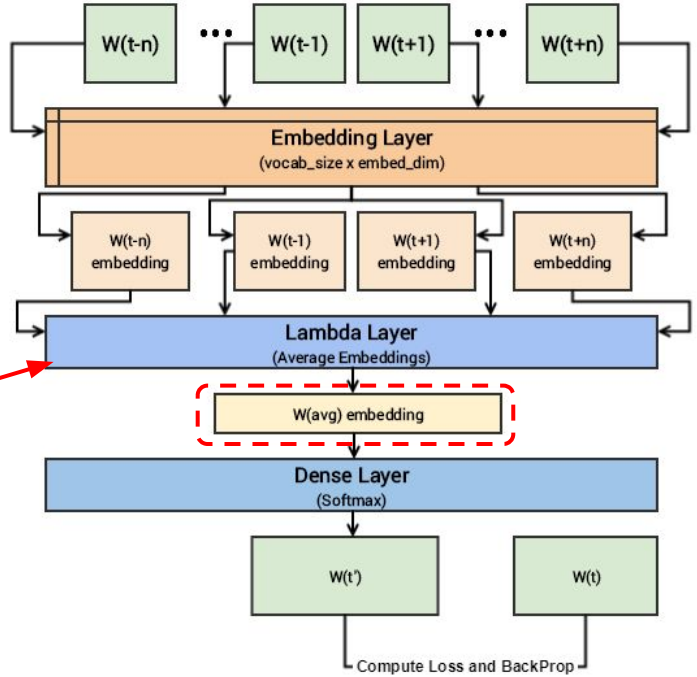


CBOW - Entrenamiento

LINK

Para entrenar necesitamos tener el vocabulario del corpus y las sentencias organizadas por el tamaño de la ventana de entrada.

Los embeddings de cada palabra son el embedding promedio de todas las veces que se utilizó en el corpus.

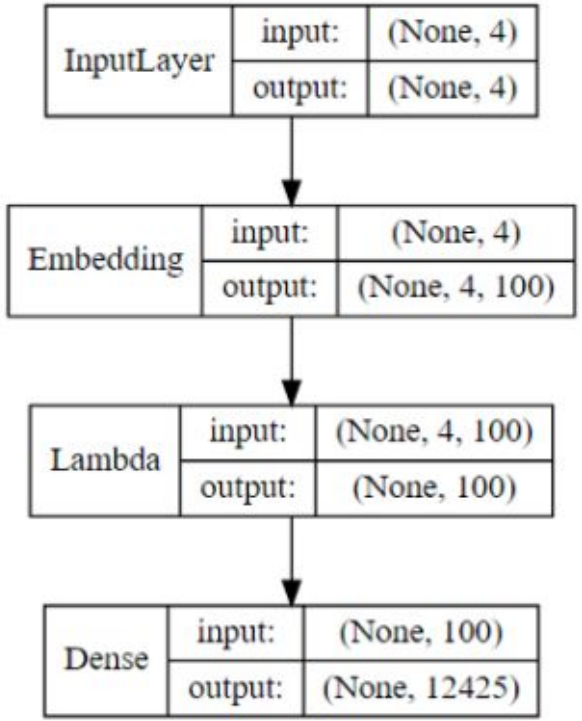



```
cbow = Sequential()
cbow.add(Embedding(input_dim=vocab_size, output_dim=embed_size, input_length=window_size*2))
cbow.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(embed_size,)))
cbow.add(Dense(vocab_size, activation='softmax'))
cbow.compile(loss='categorical_crossentropy', optimizer='rmsprop')
```

CBOW - Entrenamiento

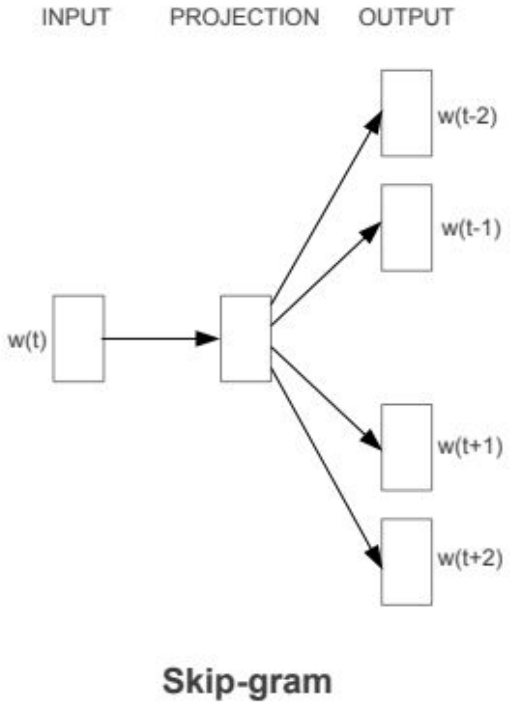
Con tan solo un corpus de 12425 palabras distintas y embedding de 100 dimensiones hay que entrenar **2.5 Millones de parámetros**

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 4, 100)            1242500
_________________________________________________________________
lambda_1 (Lambda)            (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 12425)             1254925
=================================================================
Total params: 2,497,425
Trainable params: 2,497,425
Non-trainable params: 0
_________________________________________________________________
```



Skip-Gram

Al contrario de CBOW, este modelo intenta predecir las palabras que rodean (contexto) a una palabra objetivo. Se divide el output como pares [target, context]



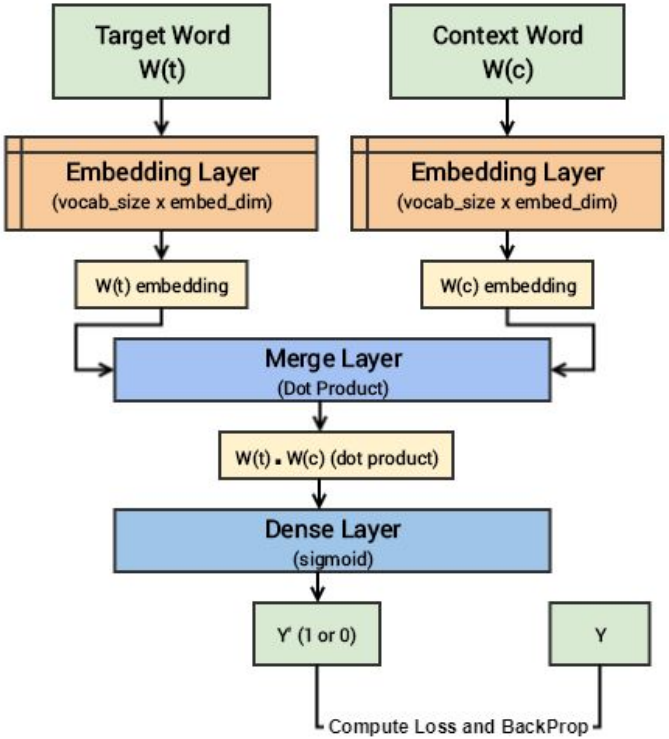
I will have orange juice and eggs for breakfast	(will, I)	(will, have)	(will, orange)	
I will have orange juice and eggs for breakfast	(have, I)	(have, will)	(have, orange)	(have, juice)
I will have orange juice and eggs for breakfast	(orange, will)	(orange, have)	(orange, juice)	(orange, and)
I will have orange juice and eggs for breakfast	(juice, have)	(juice, orange)	(juice, and)	(juice, eggs)
I will have orange juice and eggs for breakfast	(and, orange)	(and, juice)	(and, eggs)	(and, for)
I will have orange juice and eggs for breakfast	(eggs, juice)	(eggs, and)	(eggs, for)	(eggs, breakfast)
I will have orange juice and eggs for breakfast	(for, and)	(for, eggs)	(for, breakfast)	

Skip-Gram - Entrenamiento

LINK

Por cada par [target, context] el sistema determina si las palabras tiene significado en contexto (1) o no lo tiene (0), buscando así acercar las palabras que tienen significado juntas (que se espera que estén juntas en el texto)

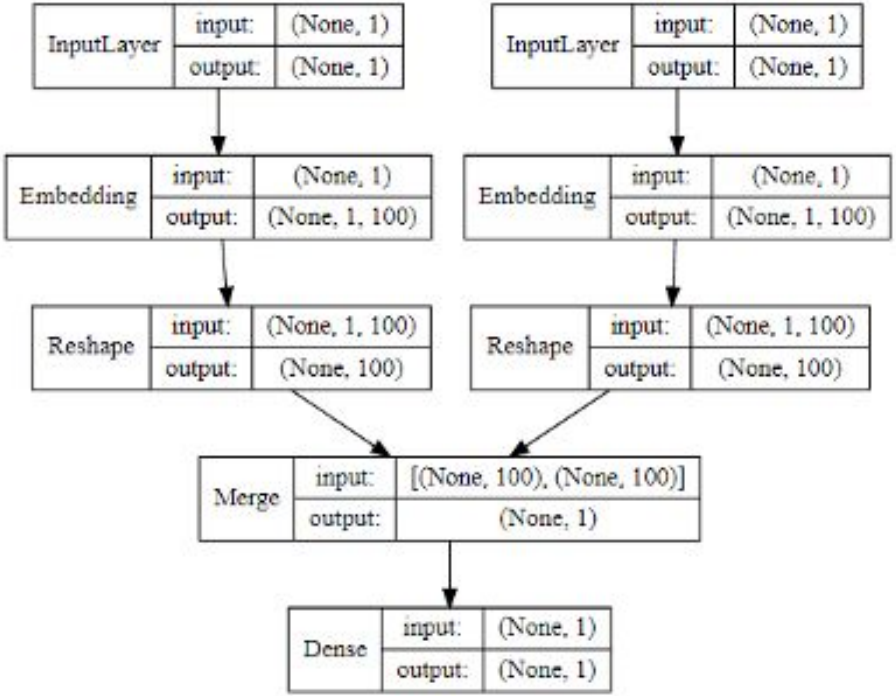
```
word_model = Sequential()
word_model.add(Embedding(vocab_size, embed_size,
                         embeddings_initializer="glorot_uniform",
                         input_length=1))
word_model.add(Reshape((embed_size, )))

context_model = Sequential()
context_model.add(Embedding(vocab_size, embed_size,
                  embeddings_initializer="glorot_uniform",
                  input_length=1))
context_model.add(Reshape((embed_size,)))

model = Sequential()
model.add(Merge([word_model, context_model], mode="dot"))
model.add(Dense(1, kernel_initializer="glorot_uniform", activation="sigmoid
```



Skip-Gram - Entrenamiento

Skip-Gram requiere más datos para lograr un buen resultado pero obtiene más información sobre el contexto del corpus en sus embeddings.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
merge_2 (Merge)              (None, 1)                 0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 2
=================================================================
Total params: 2,485,002
Trainable params: 2,485,002
Non-trainable params: 0
_________________________________________________________________
```



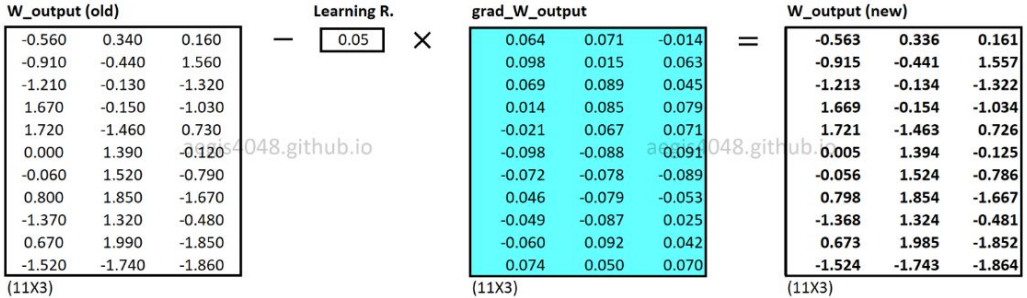
Negative sampling

LINK

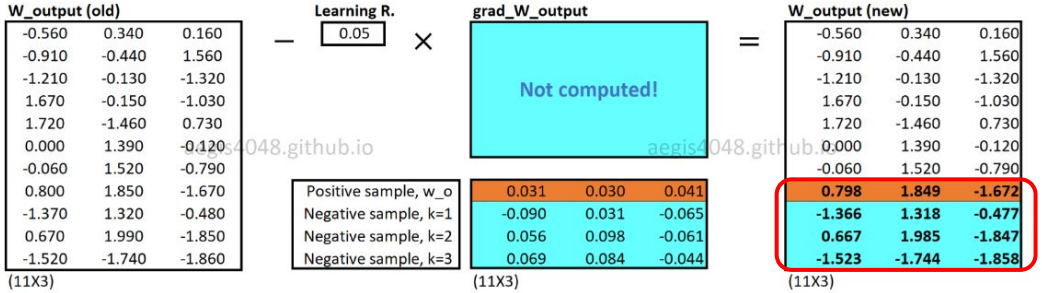
En SkipGram/CBOW la cantidad de parámetros a entrenar en la softmax es enorme:

Parametros = vocab_size * embedding_size → millones de parámetros

Vanilla Skip-Gram



Negative Sampling



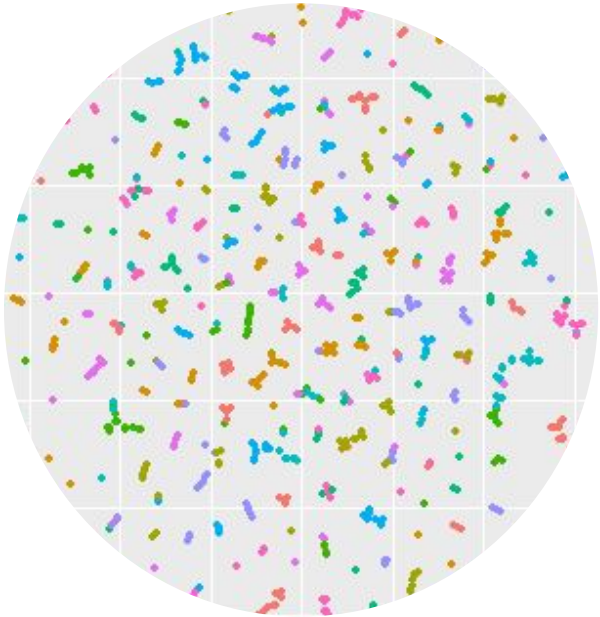
En cada iteración se observa la palabras [target, contexto] y "K" palabras aleatorias del corpus.
El objetivo es optimizar cómputo aproximando la softmax. Además funciona como regularización.
Para corpus pequeños, el muestreo debe ser mayor.

Visualizar embeddings en baja dimensionalidad:

t-SNE (t-distributed stochastic neighbor embedding)

LINK

- Técnica de reducción de dimensionalidad no-lineal (a diferencia de PCA).
- Intenta reproducir en baja dimensionalidad, la localidad de los datos en alta dimensionalidad.
- Es estocástica, a priori los resultados no se repiten.
- Por su carácter estocástico es sólo recomendable como herramienta de visualización y exploratoria.



Leer UMAP!

Gensim - Doc2Vec paragraph embeddings

LINK

Utilizaremos esta librería que nos facilita generar embeddings tipo Skip-Gram o CBOW de nuestros corpus

- Librería de Python
- Existe desde 2009 y nació originalmente para topic modelling
- Muy popular y muy simple de utilizar

Generación de embeddings con Gensim

Link al Colab

LINK

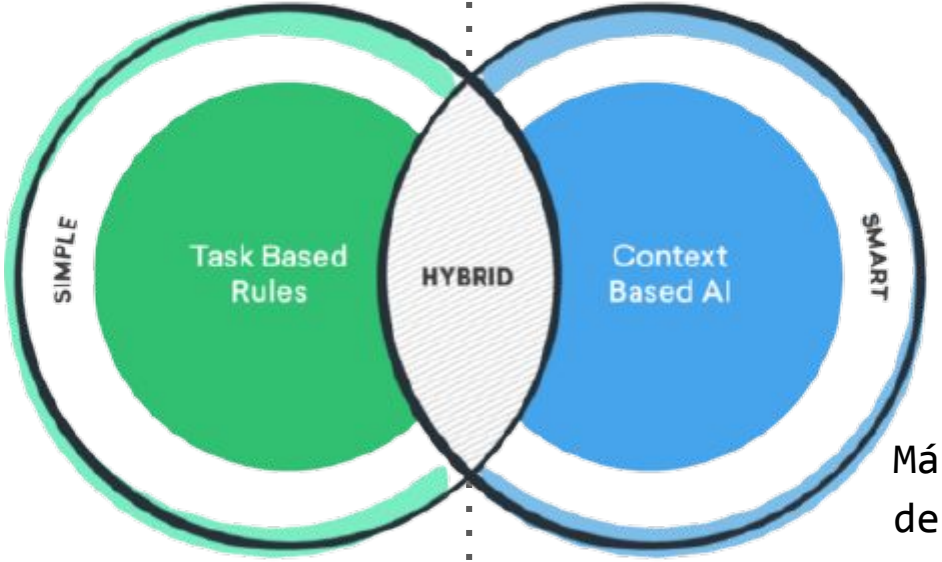
Desafío

Crear sus propios vectores con Gensim basado en lo visto en clase con otro dataset.
Probar términos de interés y explicar similitudes en el espacio de embeddings.
Intentar plantear y probar tests de analogías. Graficar los embeddings resultantes.
Sacar conclusiones.

Algunos recursos para descargar corpora de texto

Project Gutenberg	Compilación de literatura completa de dominio público principalmente en inglés.
Textos.info	Compilación de literatura completa de dominio público en español.

BOTs lingüísticos



Son limitados a una tarea específica

Fácil y "barato" de entrenar

Ideal para chats de pedidos

Interactúan casi como un humano

Más difíciles de entrenar, requieren más datos y cómputo

Ideal para asistentes virtuales

Sistema de obtención de información

Vectorizaremos texto de un corpus. Por ejemplo: párrafos u oraciones.

Vectorizaremos el texto de entrada y devolveremos la mejor coincidencia del corpus en términos de la similaridad coseno

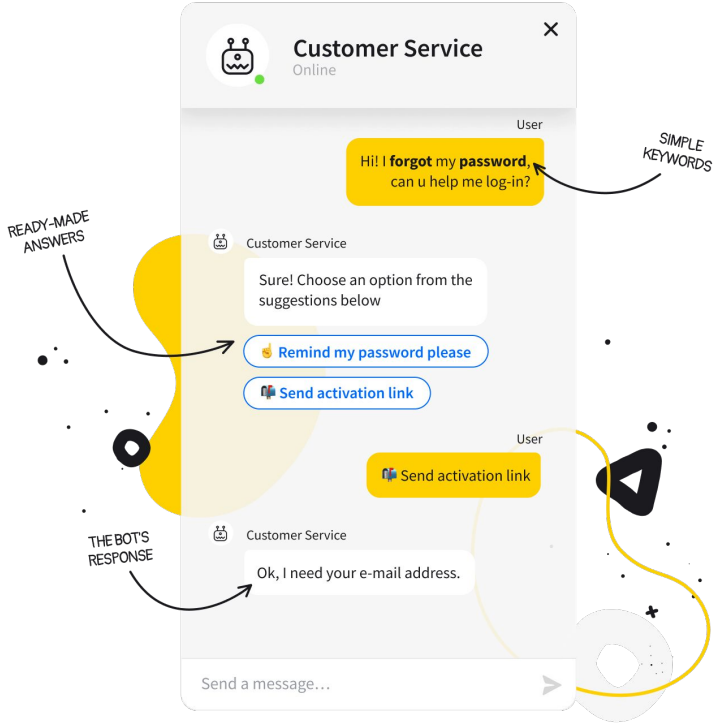
$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Probaremos el sistema armando una interfaz con gradio.

gradio

Sistema de obtención de información

BOT de consulta abierta y respuesta predeterminada



Nuestro bot será entrenado con <TAGS> (ej: saludo)

Cada <TAG> será representado por un patrón de posibles preguntas <patterns> (X)

Cada <TAG> tendrá uno o varias posibles respuestas <classes> (y)

BOT de consulta abierta

Link al Colab

LINK

Desafio

Tomar uno de los dos ejemplos mostrados y armar una versión propia

Spell checker (algunas librerias, cómo funciona), polyglot, procesar emojis, ner (dar cómo funciona).

Tokenizadores(BPE lo dejamos para después), ver si puedo simplificar la normalización de caracteres raros.

Meter vectorizadores de sklearn, algo de scrapping (no puede ser muy complejo).