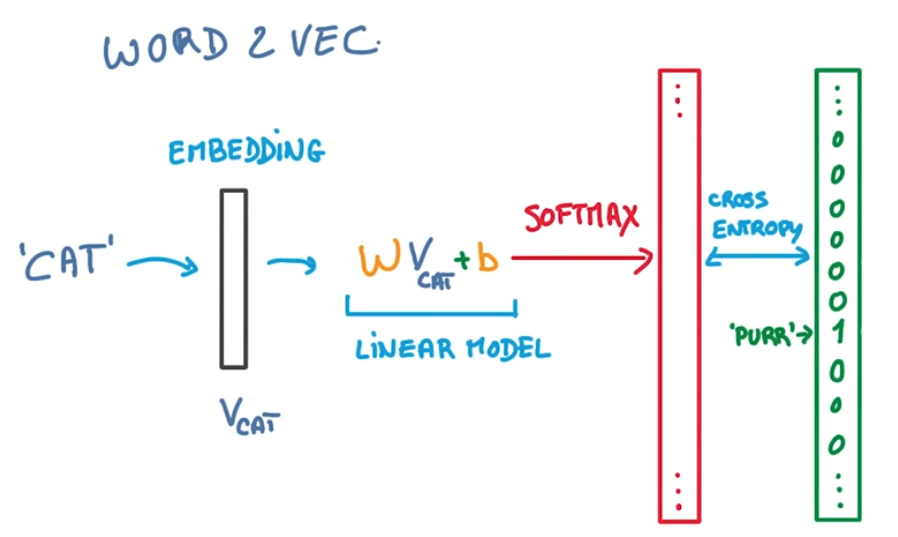
Word2Vec CON Gensim - Python



Adrián Yared Armas de la Nuez

**Contenido**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

[**1. Enunciado 2**](#_2c35568x6wg8)

[**2.1 Word2Vec model trainning 2**](#_t8cwbo6n4466)

[**2.1.1 Install gensim 2**](#_aig92oe7k2m0)

[**2.1.2 Ejecución 2**](#_e9uqlcl56mu)

[**2.1.3 Imports 2**](#_6f6pabepqbvr)

[**2.1.4 Import dataset 2**](#_mf155rww6l6g)

[**2.1.5 Ejecución 2**](#_2acd74hu073f)

[**2.1.6 Mostrar las primeras 10 filas 2**](#_h9dcf6q2svmy)

[**2.1.7 Ejecución 3**](#_sz8tcjhj2hyh)

[**2.1.8 Mostrar los títulos 3**](#_e74z2lm9c1sn)

[**2.1.9 Ejecución 3**](#_b9sd8zy32n1)

[**2.1.10 Prepare data 3**](#_7y4ryilfhc4y)

[**2.1.10.1 Install punkt\_tab 3**](#_hyl50rokgggn)

[**2.1.10.1 Tokenización de palabras 4**](#_9kuruukq59i)

[**2.1.10.1 Tokenización de palabras 4**](#_9akzu5345ff7)

[**2.1.11 Entrenamiento del modelo Word2Vec 4**](#_iy6apbqdzo4n)

[**2.1.12 Explorar el modelo 4**](#_f14jg744yc2m)

[**2.2 Usando el modelo pre-entrenado word2vec 5**](#_lbyhbgewvh8g)

[**2.2.1 Install gdown 5**](#_2vaiciplhxye)

[**2.2.2 Descarga del modelo 6**](#_agdf8jbr0shz)

[**2.2.3 Imports 6**](#_38zeu7ez0yyc)

[**2.2.4 Carga del modelo preentrenado 6**](#_kwch9by5hk1y)

[**2.2.5 Exploración del modelo preentrenado 6**](#_bhc7ipkriuhe)

[**2.2.6 Vector similar a queen 7**](#_hrumohhopxp0)

[**2.2.7 Otros ejemplos 7**](#_jx0dpt9xqqmc)

[**3. Github y Colab 8**](#_lmdngf8dlin3)

## **1. Enunciado**

En el siguiente link <https://www.youtube.com/watch?v=Z1VsHYcNXDI> puedes acceder

al vídeo explicativo del uso del algoritmo Word2Vec con Gensim en Python.

Implementa el código, añade celdas markdown con los comentarios y explicaciones

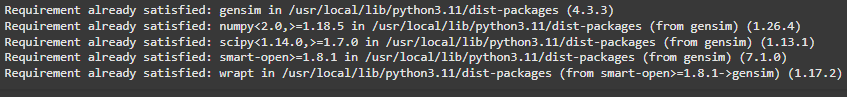
oportunas.

### **2.1 Word2Vec model trainning**

#### **2.1.1 Install gensim**

pip install gensim

#### **2.1.2 Ejecución**



#### **2.1.3 Imports**

from gensim.models import Word2Vec,keyedvectors # Does topic modeling and document similarity

import pandas as pd # Does data manipulation and analysis

import nltk # Its a natural Language Processing toolkit.

import kagglehub # Kaggle download library

#### **2.1.4 Import dataset**

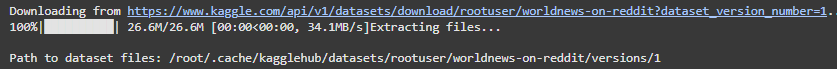
# Download latest version

path = kagglehub.dataset\_download("rootuser/worldnews-on-reddit")

csvPath = path + "/reddit\_worldnews\_start\_to\_2016-11-22.csv"

print("Path to dataset files:", path)

#### **2.1.5 Ejecución**



#### **2.1.6 Mostrar las primeras 10 filas**

df = pd.read\_csv(csvPath)

df.head(10)

#### **2.1.7 Ejecución**



#### **2.1.8 Mostrar los títulos**

# Get all title values

newsTitles = df["title"].values

newsTitles

#### **2.1.9 Ejecución**

### 

#### **2.1.10 Prepare data**

##### **2.1.10.1 Install punkt\_tab**

#only once

nltk.download('punkt\_tab')

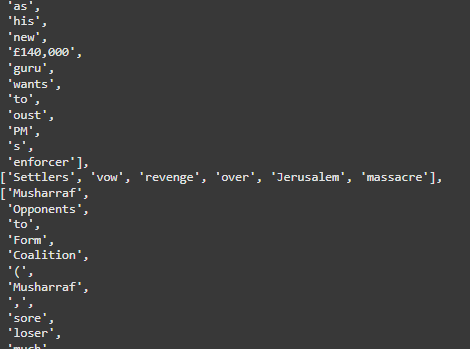
##### **2.1.10.1 Tokenización de palabras**

newsVec = [nltk.word\_tokenize(title) for title in newsTitles]

##### **2.1.10.1 Tokenización de palabras**

# Show all vec

newsVec



#### **2.1.11 Entrenamiento del modelo Word2Vec**

# Trains a Word2Vec model on the tokenized news titles. min\_count=1 means even words that appear

# only once are considered, and vector\_size=32 sets the dimensionality of the word vectors.

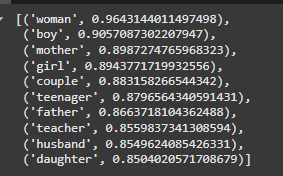
model = Word2Vec(newsVec, min\_count=1, vector\_size=32)

#### **2.1.12 Explorar el modelo**

Vectores similares a man:

# Find similar vectorized words to 'man'

model.wv.most\_similar('man')



Vectores similares a queen:

# Finds words most similar to "man", performs vector arithmetic (king - man + woman),

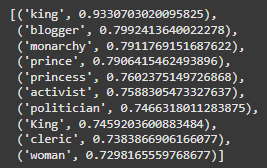
# and finds the most similar word to the resulting vector (likely "queen").

vec = model.wv['king'] - model.wv['man'] + model.wv['woman']

# This output is a list of tuples, where each tuple represents a word and its similarity

# score to the target vector (vec, which was calculated as king - man + woman).

model.wv.most\_similar([vec])

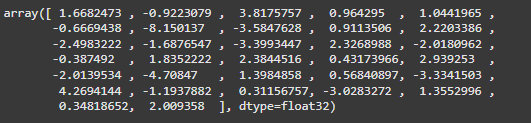


Similar al vector man:

# This output is a list of tuples, where each tuple represents a word and its similarity

# score to the target vector (man).

model.wv['man']

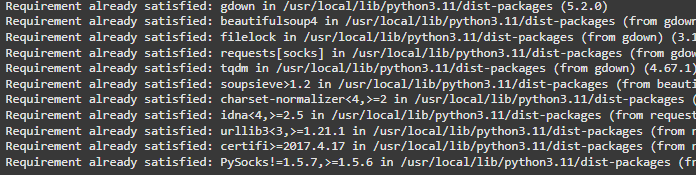


### **2.2 Usando el modelo pre-entrenado word2vec**

#### **2.2.1 Install gdown**

# Gdown downloads files from Google Drive.

pip install gdown



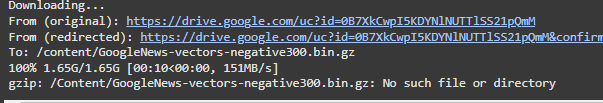
#### **2.2.2 Descarga del modelo**

#Download the pre-trained model

!gdown 0B7XkCwpI5KDYNlNUTTlSS21pQmM -O GoogleNews-vectors-negative300.bin.gz

# Unzip

!gzip -d /content/GoogleNews-vectors-negative300.bin.gz



#### **2.2.3 Imports**

from gensim.models import Word2Vec,keyedvectors # Does topic modeling and document similarity

import pandas as pd # Does data manipulation and analysis

import nltk # Its a natural Language Processing toolkit.

#### **2.2.4 Carga del modelo preentrenado**

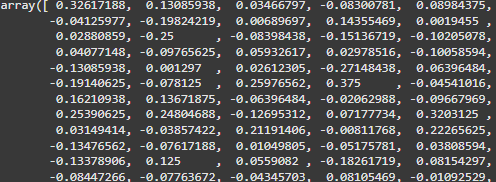
# Loads a pre-trained Word2Vec model into a variable named model. model = GoogleNews-vectors-negative300

model = KeyedVectors.load\_word2vec\_format('GoogleNews-vectors-negative300.bin', binary=True, limit=100000)

#### **2.2.5 Exploración del modelo preentrenado**

# print all vec

model["man"]



#### **2.2.6 Vector similar a queen**

# Finds words most similar to "man", performs vector arithmetic (king - man + woman),

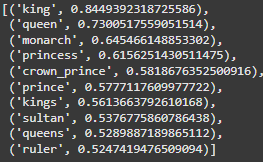
# and finds the most similar word to the resulting vector (likely "queen").

vec = model ["king"] - model["man"] + model["woman"]

# This output is a list of tuples, where each tuple represents a word and its similarity

# score to the target vector.

model.most\_similar([vec])



#### **2.2.7 Otros ejemplos**

Francia:

# The output of model.most\_similar([vec]) would likely be a list of words (countries) where "France" is at or near the top, along with possibly other

# countries that have a similar relationship to their capitals as the Germany-Berlin relationship. This is because the vector arithmetic captures the

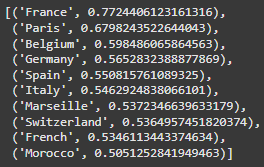
# "capital city of" relationship

vec = model ["Germany"] - model["Berlin"] + model["Paris"]

model.most\_similar([vec])

# This output would indicate that "France" is the most similar word to the vector you calculated,

# followed by "Paris," "Belgium," and so on. The numbers represent the similarity scores.



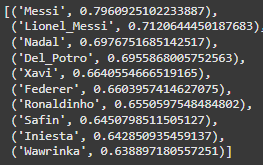
Fútbol:

# The code uses vector arithmetic to find the analogy between Messi and football and applies it to tennis to find a similar entity. The output suggests that the model successfully identifies famous

# tennis players like Nadal and Federer, showcasing how word embeddings can capture relationships and analogies between words and concepts. I hope this helps! Let me know if you have any other questions.

vec = model ["Messi"] - model["football"] + model["tennis"]

model.most\_similar([vec])



## **3. Github y Colab**

[](https://colab.research.google.com/drive/18JNNFLDRILCNOfyDCcxyIQqdYLWFZWdV#scrollTo=H3oaeVsvySnj)