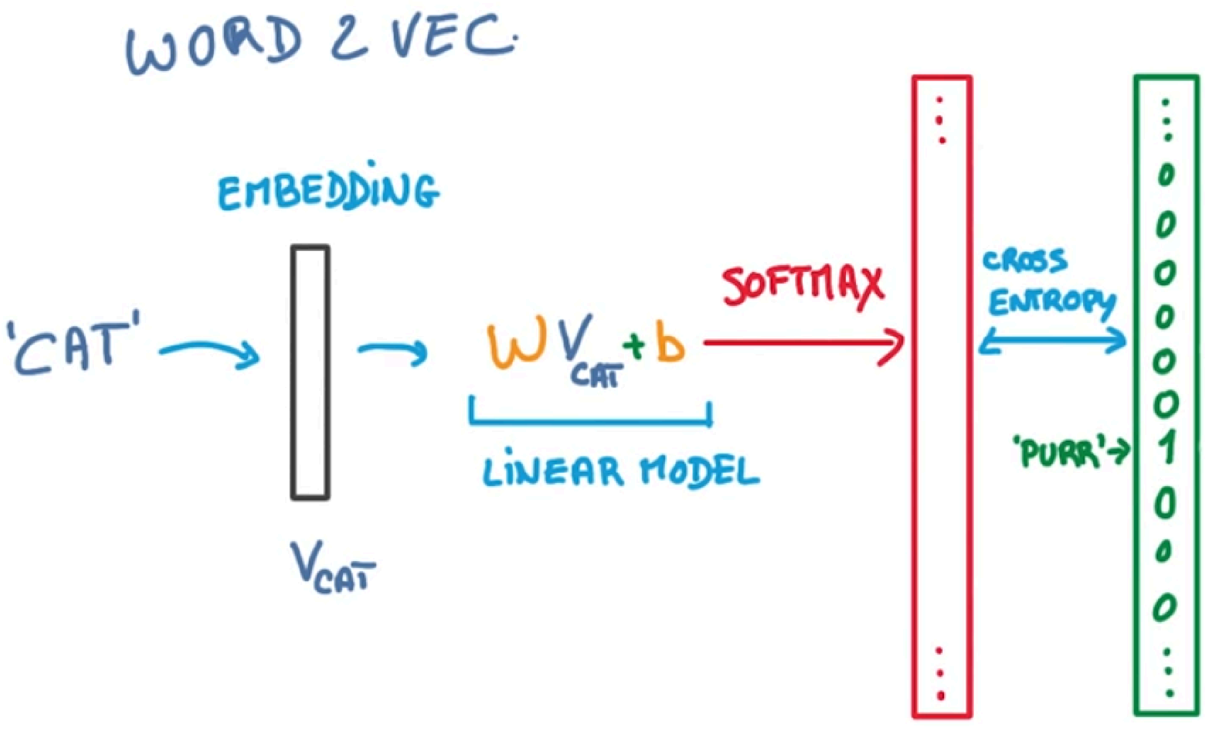
Word2Vec CON Gensim - Python



Adrián Yared Armas de la Nuez

Contenido

1. Enunciado.......... 2
   2.1 Word2Vec model trainning.......... 2
      2.1.1 Install gensim.......... 2
      2.1.2 Ejecución.......... 2
      2.1.3 Imports.......... 2
      2.1.4 Import dataset.......... 2
      2.1.5 Ejecución.......... 2
      2.1.6 Mostrar las primeras 10 filas.......... 2
      2.1.7 Ejecución.......... 3
      2.1.8 Mostrar los títulos.......... 3
      2.1.9 Ejecución.......... 3
      2.1.10 Prepare data.......... 3
         2.1.10.1 Install punkt_tab.......... 3
         2.1.10.1 Tokenización de palabras.......... 4
         2.1.10.1 Tokenización de palabras.......... 4
      2.1.11 Entrenamiento del modelo Word2Vec.......... 4
      2.1.12 Explorar el modelo.......... 4
   2.2 Usando el modelo pre-entrenado word2vec.......... 5
      2.2.1 Install gdown.......... 5
      2.2.2 Descarga del modelo.......... 6
      2.2.3 Imports.......... 6
      2.2.4 Carga del modelo preentrenado.......... 6
      2.2.5 Exploración del modelo preentrenado.......... 6
      2.2.6 Vector similar a queen.......... 7
      2.2.7 Otros ejemplos.......... 7
3. Github y Colab.......... 8

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.

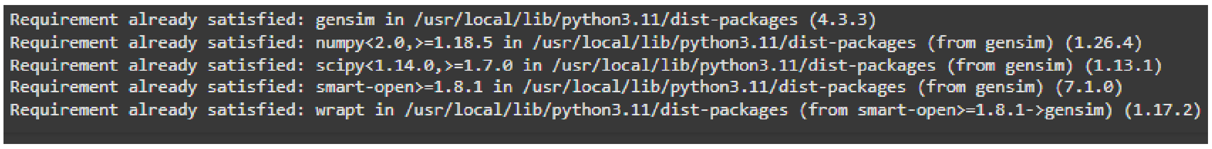
Implementa el código, añade celdas markdown con los comentarios y explicaciones oportunas.

2.1 Word2Vec model trainning

2.1.1 Install gensim

```
pip install gensim
```

2.1.2 Ejecución



```
Requirement already satisfied: gensim in /usr/local/lib/python3.11/dist-packages (4.3.3)
Requirement already satisfied: numpy<2.0,>=1.18.5 in /usr/local/lib/python3.11/dist-packages (from gensim) (1.26.4)
Requirement already satisfied: scipy<1.14.0,>=1.7.0 in /usr/local/lib/python3.11/dist-packages (from gensim) (1.13.1)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.11/dist-packages (from gensim) (7.1.0)
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open>=1.8.1->gensim) (1.17.2)
```

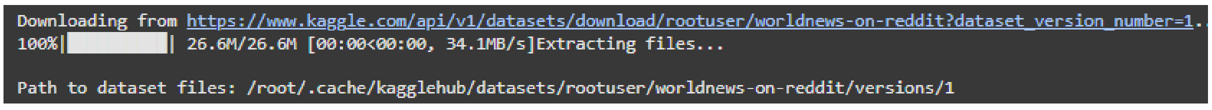
2.1.3 Imports

```
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



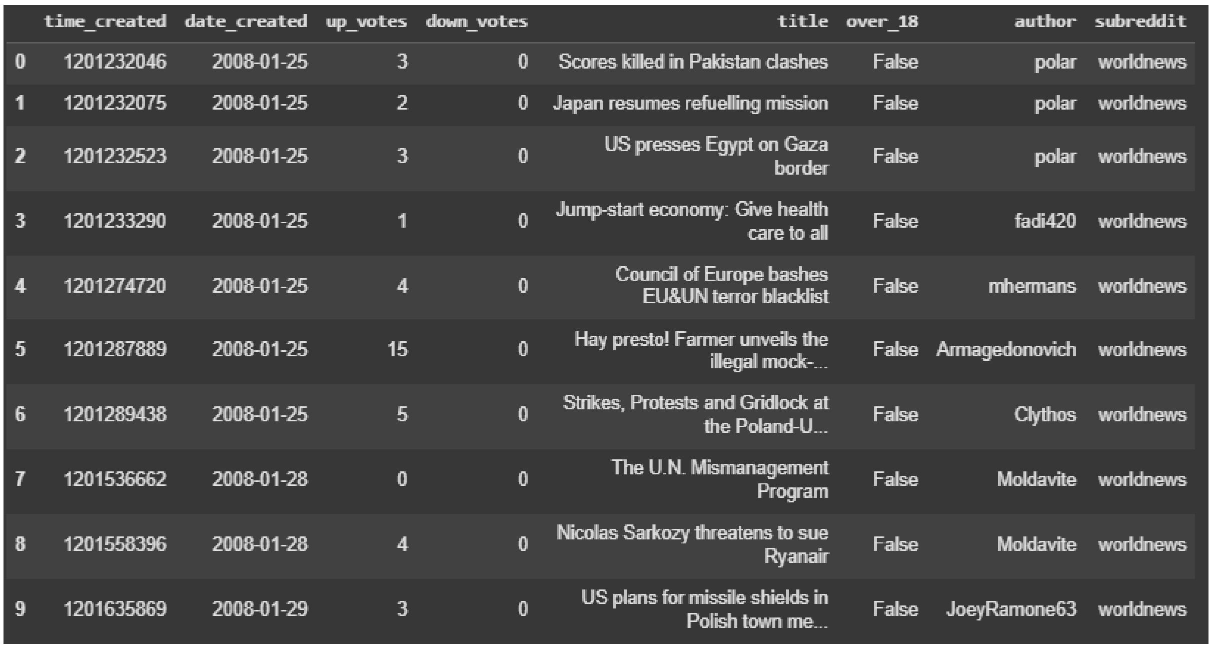
```
Downloading from https://www.kaggle.com/api/v1/datasets/download/rootuser/worldnews-on-reddit?dataset_version_number=1..
100%|██████████| 26.6M/26.6M [00:00<00:00, 34.1MB/s]Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/rootuser/worldnews-on-reddit/versions/1
```

2.1.6 Mostrar las primeras 10 filas

```
df = pd.read_csv(csvPath)
df.head(10)
```

2.1.7 Ejecución

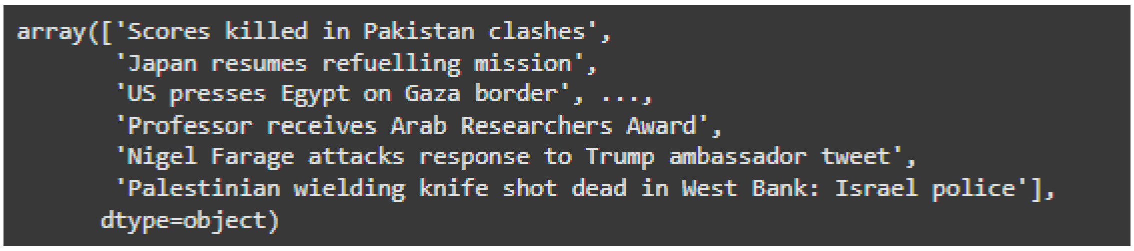


	time_created	date_created	up_votes	down_votes	title	over_18	author	subreddit
0	1201232046	2008-01-25	3	0	Scores killed in Pakistan clashes	False	polar	worldnews
1	1201232075	2008-01-25	2	0	Japan resumes refuelling mission	False	polar	worldnews
2	1201232523	2008-01-25	3	0	US presses Egypt on Gaza border	False	polar	worldnews
3	1201233290	2008-01-25	1	0	Jump-start economy: Give health care to all	False	fadi420	worldnews
4	1201274720	2008-01-25	4	0	Council of Europe bashes EU&UN terror blacklist	False	mhermans	worldnews
5	1201287889	2008-01-25	15	0	Hay presto! Farmer unveils the illegal mock-...	False	Armagedonovich	worldnews
6	1201289438	2008-01-25	5	0	Strikes, Protests and Gridlock at the Poland-U...	False	Clythos	worldnews
7	1201536662	2008-01-28	0	0	The U.N. Mismanagement Program	False	Moldavite	worldnews
8	1201558396	2008-01-28	4	0	Nicolas Sarkozy threatens to sue Ryanair	False	Moldavite	worldnews
9	1201635869	2008-01-29	3	0	US plans for missile shields in Polish town me...	False	JoeyRamone63	worldnews

2.1.8 Mostrar los títulos

```
# Get all title values
newsTitles = df["title"].values
newsTitles
```

2.1.9 Ejecución



```
array(['Scores killed in Pakistan clashes',
       'Japan resumes refuelling mission',
       'US presses Egypt on Gaza border', ...,
       'Professor receives Arab Researchers Award',
       'Nigel Farage attacks response to Trump ambassador tweet',
       'Palestinian wielding knife shot dead in West Bank: Israel police'],
      dtype=object)
```

2.1.10 Prepare data

2.1.10.1 Install punkt_tab

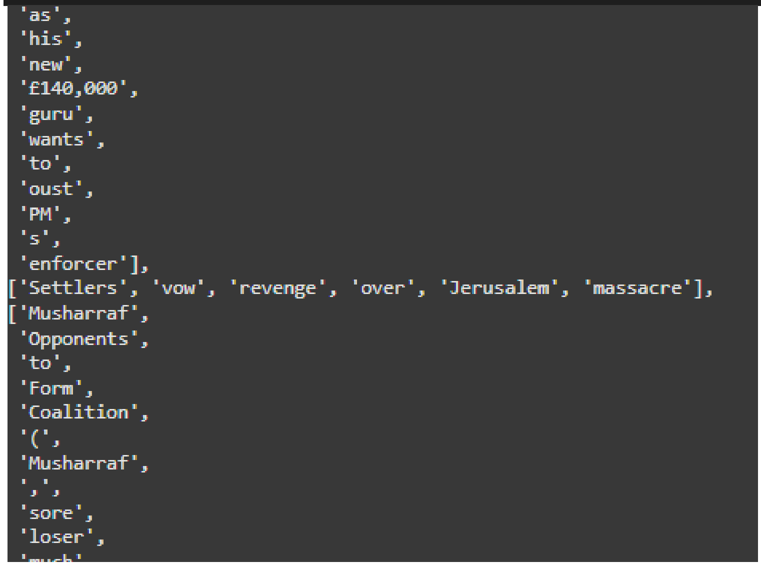
```
#only once
nltk.download('punkt_tab')
```

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
```



```
 'as',
 'his',
 'new',
 '£140,000',
 'guru',
 'wants',
 'to',
 'oust',
 'PM',
 's',
 'enforcer'],
['Settlers', 'vow', 'revenge', 'over', 'Jerusalem', 'massacre'],
['Musharraf',
 'Opponents',
 'to',
 'Form',
 'Coalition',
 '(',
 'Musharraf',
 ',',
 'sore',
 'loser',
```

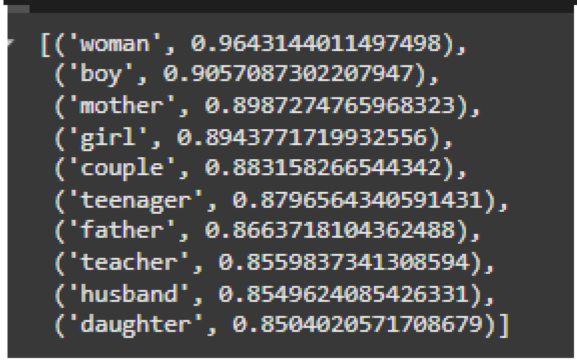
2.1.11 Entrenamiento del modelo Word2Vec

```
# 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:

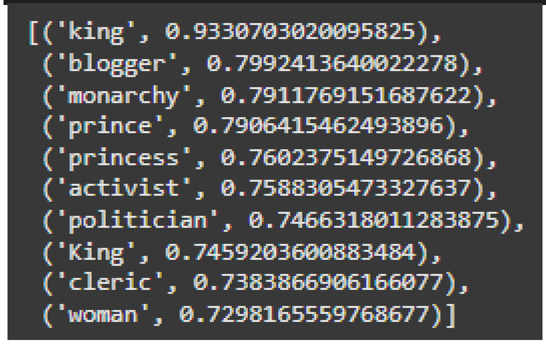
```
# Find similar vectorized words to 'man'
model.wv.most_similar('man')
```



```
[('woman', 0.9643144011497498),
 ('boy', 0.9057087302207947),
 ('mother', 0.8987274765968323),
 ('girl', 0.8943771719932556),
 ('couple', 0.883158266544342),
 ('teenager', 0.8796564340591431),
 ('father', 0.8663718104362488),
 ('teacher', 0.8559837341308594),
 ('husband', 0.8549624085426331),
 ('daughter', 0.8504020571708679)]
```

Vectores similares a queen:

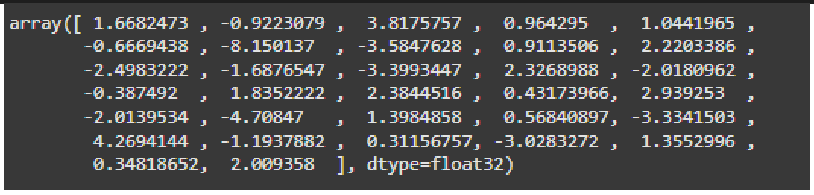
```
# Finds words most similar to "man", performs vector arithmetic (king -
man + woman),
# and finds the most similar word to the resulting vector (likely
"queen").
vec = model.wv['king'] - model.wv['man'] + model.wv['woman']
# This output is a list of tuples, where each tuple represents a word
and its similarity
# score to the target vector (vec, which was calculated as king - man +
woman).
model.wv.most_similar([vec])
```



```
[('king', 0.9330703020095825),
 ('blogger', 0.7992413640022278),
 ('monarchy', 0.7911769151687622),
 ('prince', 0.7906415462493896),
 ('princess', 0.7602375149726868),
 ('activist', 0.7588305473327637),
 ('politician', 0.7466318011283875),
 ('King', 0.7459203600883484),
 ('cleric', 0.7383866906166077),
 ('woman', 0.7298165559768677)]
```

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']
```

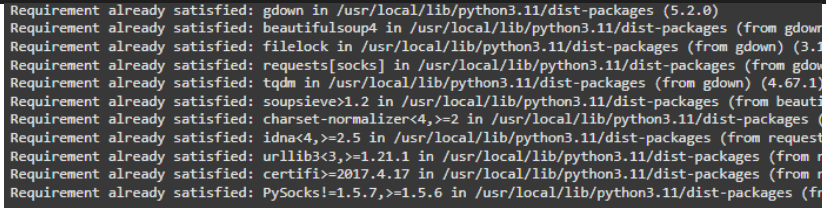


```
array([ 1.6682473 , -0.9223079 ,  3.8175757 ,  0.964295  ,  1.0441965 ,
       -0.6669438 , -8.150137  , -3.5847628 ,  0.9113506 ,  2.2203386 ,
       -2.4983222 , -1.6876547 , -3.3993447 ,  2.3268988 , -2.0180962 ,
       -0.387492  ,  1.8352222 ,  2.3844516 ,  0.43173966,  2.939253  ,
       -2.0139534 , -4.70847   ,  1.3984858 ,  0.56840897, -3.3341503 ,
        4.2694144 , -1.1937882 ,  0.31156757, -3.0283272 ,  1.3552996 ,
        0.34818652,  2.009358  ], dtype=float32)
```

2.2 Usando el modelo pre-entrenado word2vec

2.2.1 Install gdown

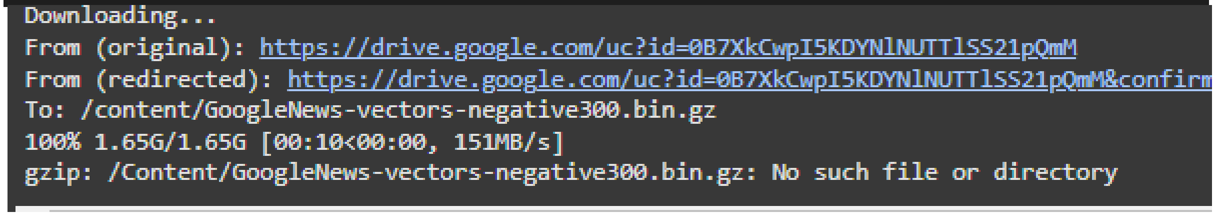
```
# Gdown downloads files from Google Drive.
pip install gdown
```



```
Requirement already satisfied: gdown in /usr/local/lib/python3.11/dist-packages (5.2.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.11/dist-packages (from gdow
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from gdown) (3.1
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.11/dist-packages (from gdo
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from gdown) (4.67.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packages (from beauti
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from request
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from r
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from r
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.11/dist-packages (fr
```

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
```



```
Downloading...
From (original): https://drive.google.com/uc?id=0B7XkCwpI5KDYNlNUTTlSS21pQmM
From (redirected): https://drive.google.com/uc?id=0B7XkCwpI5KDYNlNUTTlSS21pQmM&confirm
To: /content/GoogleNews-vectors-negative300.bin.gz
100% 1.65G/1.65G [00:10<00:00, 151MB/s]
gzip: /Content/GoogleNews-vectors-negative300.bin.gz: No such file or directory
```

2.2.3 Imports

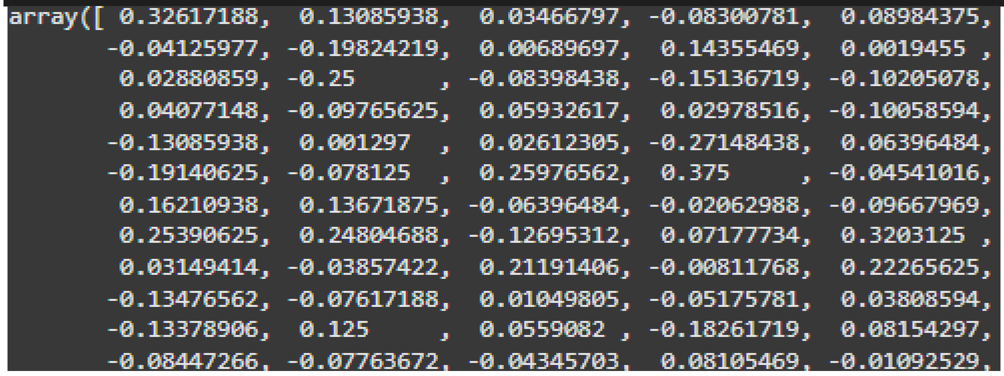
```
from gensim.models import Word2Vec,keyedvectors # Does topic modeling
and document similarity
import pandas as pd # Does data manipulation and analysis
import nltk # Its a natural Language Processing toolkit.
```

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

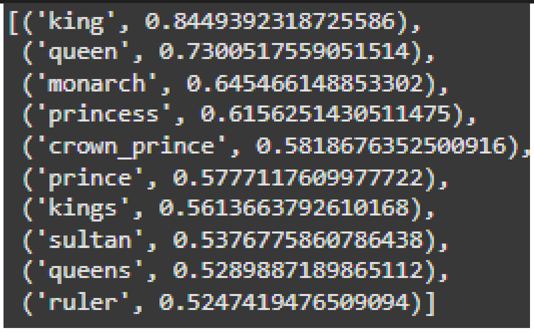
```
# print all vec
model["man"]
```



```
array([ 0.32617188,  0.13085938,  0.03466797, -0.08300781,  0.08984375,
       -0.04125977, -0.19824219,  0.00689697,  0.14355469,  0.0019455 ,
        0.02880859, -0.25      , -0.08398438, -0.15136719, -0.10205078,
        0.04077148, -0.09765625,  0.05932617,  0.02978516, -0.10058594,
       -0.13085938,  0.001297  ,  0.02612305, -0.27148438,  0.06396484,
       -0.19140625, -0.078125  ,  0.25976562,  0.375     , -0.04541016,
        0.16210938,  0.13671875, -0.06396484, -0.02062988, -0.09667969,
        0.25390625,  0.24804688, -0.12695312,  0.07177734,  0.3203125 ,
        0.03149414, -0.03857422,  0.21191406, -0.00811768,  0.22265625,
       -0.13476562, -0.07617188,  0.01049805, -0.05175781,  0.03808594,
       -0.13378906,  0.125     ,  0.0559082 , -0.18261719,  0.08154297,
       -0.08447266, -0.07763672, -0.04345703,  0.08105469, -0.01092529,
```

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])
```

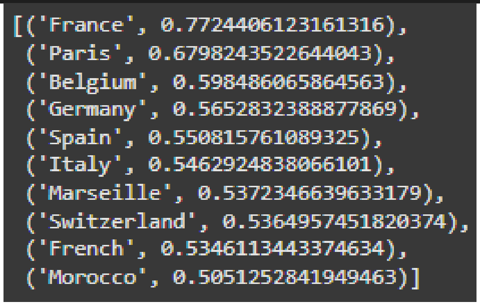


```
[('king', 0.8449392318725586),
 ('queen', 0.7300517559051514),
 ('monarch', 0.645466148853302),
 ('princess', 0.6156251430511475),
 ('crown_prince', 0.5818676352500916),
 ('prince', 0.5777117609977722),
 ('kings', 0.5613663792610168),
 ('sultan', 0.5376775860786438),
 ('queens', 0.5289887189865112),
 ('ruler', 0.5247419476509094)]
```

2.2.7 Otros ejemplos

Francia:

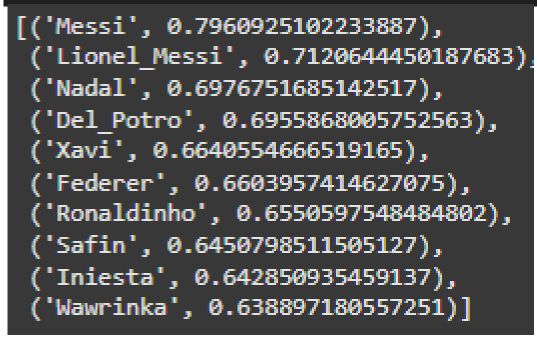
```
# The output of model.most_similar([vec]) would likely be a list of
words (countries) where "France" is at or near the top, along with
possibly other
# countries that have a similar relationship to their capitals as the
Germany-Berlin relationship. This is because the vector arithmetic
captures the
# "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.
```



```
[('France', 0.7724406123161316),
 ('Paris', 0.6798243522644043),
 ('Belgium', 0.598486065864563),
 ('Germany', 0.5652832388877869),
 ('Spain', 0.550815761089325),
 ('Italy', 0.5462924838066101),
 ('Marseille', 0.5372346639633179),
 ('Switzerland', 0.5364957451820374),
 ('French', 0.5346113443374634),
 ('Morocco', 0.5051252841949463)]
```

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])
```



```
[('Messi', 0.7960925102233887),
 ('Lionel_Messi', 0.7120644450187683),
 ('Nadal', 0.6976751685142517),
 ('Del_Potro', 0.6955868005752563),
 ('Xavi', 0.6640554666519165),
 ('Federer', 0.6603957414627075),
 ('Ronaldinho', 0.6550597548484802),
 ('Safin', 0.6450798511505127),
 ('Iniesta', 0.642850935459137),
 ('Wawrinka', 0.638897180557251)]
```

3. Github y Colab