

UANL  
Maestría en Ciencia de Datos  
Procesamiento y clasificación de Datos  
Alder López Cerda

**Algoritmo de comparación de coches basado redes neuronales y procesamiento natural de lenguaje.**

## **Resumen**

*Uso de redes neuronales basado en procesamiento natural de lenguaje para la comparativa entre coches para determinar su grado de similitud y además encontrar coches similares.*

## **Introducción**

*En la actualidad cuando una persona busca comprar un coche se enfrenta a diferentes retos, primero encontrar un coche que le convenza y le guste, una vez que le gusto un tipo de coche se detona la necesidad de comparar ese coche con otras marcas con un coche del mismo tipo o segmento. Sucede frecuentemente que el usuario no sabe con qué comparar, ejemplo si me gusto un Mazda 3, con que coche lo comparo de Honda, Ford, Hyundai, etc. Al seleccionar un coche ¿por dónde empezar a comparar?, ¿necesito ser experto en coches para entender las diferencias entre uno y otro? o ¿saber que tanto se parecen?, la respuesta es que no ya que el planteamiento de este documento es mostrar como al seleccionar un coche comparar con otro y ver que tan similares son, o poder identificar los más similares.*

Muchos sistemas y técnicas actuales de **Procesamiento Natural de Lenguaje (PNL)** tratan las palabras como unidades atómicas: no existe la noción de similitud entre las palabras, ya que se representan como índices en un vocabulario. Esta elección tiene varias buenas razones: simplicidad, robustez y la observación de que los modelos simples entrenados con grandes cantidades de datos superan a los sistemas complejos entrenados con menos datos. Un ejemplo es el popular modelo de N-gramas utilizado para el modelado de lenguaje estadístico; hoy en día, es posible entrenar N-gramas en prácticamente todos los datos disponibles (billones de palabras). Con el progreso de las técnicas de aprendizaje automático en los últimos años, se ha vuelto posible entrenar modelos más complejos en conjuntos de datos mucho más grandes y, por lo general, superan a los modelos simples. Probablemente el concepto más exitoso es el uso de representaciones distribuidas de palabras. Por ejemplo, los modelos de lenguaje basados en redes neuronales superan significativamente a los modelos de N-gramas.

## **Objetivos del artículo**

- Desarrollar un **algoritmo para realizar comparación** de un coche con otros, así como encontrar similares.
- Abordar la base analítica matemática usando **Word2Vec** que se basa en distancia euclidiana y similitud por coseno usando la técnica para aprender vectores de palabras de alta calidad a partir de grandes conjuntos de datos con miles de millones de palabras y con millones de palabras en el vocabulario.

## Metodología

Existen diferentes tipos de modelos para estimar representaciones continuas de palabras, como el conocido *Análisis Semántico Latente (LSA)* y la *Asignación de Dirichlet Latente (LDA)*. En este caso partiremos de las representaciones distribuidas de palabras aprendidas por redes neuronales, ya que se está demostrado que funcionan significativamente mejor que LSA para preservar regularidades lineales entre palabras; Además, LDA se vuelve muy costoso desde el punto de vista computacional en grandes conjuntos de datos.

Para comparar la calidad de las diferentes versiones de los vectores de palabras, se suelen usar una tabla que muestra palabras de ejemplo y sus palabras más similares, y las entienden intuitivamente. Aunque es fácil demostrar que la palabra Francia es similar a Italia y quizás a algunos otros países, es mucho más desafiante cuando se someten esos vectores a una tarea de similitud más compleja; por ejemplo, la palabra grande es similar a más grande en el mismo sentido que pequeño es similar a más pequeño. Ejemplo de otro tipo de relación pueden ser los pares de palabras grande - más grande y pequeño - más pequeño. Además, denotamos dos pares de palabras con la misma relación como una pregunta, ya que podemos preguntar: "¿Cuál es la palabra que es similar a pequeño en el mismo sentido que el más grande es similar a grande?".

Cuando entrenamos vectores de palabras de alta dimensión en una gran cantidad de datos, los vectores resultantes se pueden usar para responder a relaciones semánticas muy sutiles entre palabras, como una ciudad y el país al que pertenece, p. Francia es a París lo que Alemania es a Berlín. Los vectores de palabras con tales relaciones semánticas podrían usarse para mejorar muchas aplicaciones de PNL existentes, como la traducción automática, la recuperación de información y los sistemas de respuesta a preguntas.

Partiendo de lo anterior, utilizaremos la similitud entre las características de dos coches o encontrar coches similares usaremos una técnica usando Procesamiento Natural de Lenguaje. *El algoritmo Word2vec utiliza un modelo de red neuronal para aprender asociaciones de palabras a partir de un gran corpus de texto.* Una vez entrenado, dicho modelo puede detectar palabras sinónimas o sugerir palabras adicionales para una frase sin terminar. Word2vec representa cada palabra distinta con una lista particular de números llamada vector. Los vectores están seleccionados cuidadosamente de forma que una función matemática (similitud coseno entre los vectores) indica el nivel de la similitud semántica entre las palabras representada por dichos vectores.

**Word2Vec** es un grupo de modelos que se utilizan para producir word embeddings. Estos modelos son poco profundos, se entrenan redes neuronales de dos capas para reconstruir contextos lingüísticos de palabras. Word2vec toma como entrada un gran corpus de texto y produce un espacio vectorial, típicamente de varios cientos de dimensiones, asignando cada palabra única en el corpus a un vector correspondiente en el espacio. Los word embeddings están colocados en el espacio vectorial de forma que las palabras que comparten contextos comunes en el corpus están localizadas cerca unas de otras en el espacio.

El Word Embedding designa un conjunto de métodos de aprendizaje que pretende representar las palabras de un texto mediante vectores de números reales. Word Embedding es capaz de capturar el contexto, la similitud semántica y sintáctica (género, sinónimos, etc.) de una palabra reduciendo su dimensión. El método de embedding que se usa habitualmente para reducir la dimensión de un vector consiste en utilizar el resultado que devuelve una capa densa, es decir, multiplicar una matriz de embedding  $W$  por la representación one hot de la palabra:

$$h_i = \sum_{j=1}^{N_{dim}} w_{ij} \cdot x_j \quad \Longleftrightarrow \quad \mathbf{h} = \mathbf{W} \cdot \mathbf{X}$$

Existen dos variantes de Word2vec, ambas utilizan una red neuronal de 3 capas (1 capa de entrada, 1 capa oculta, 1 capa de salida): Common Bag Of Words (CBOW) y Skip-gram.

Por ejemplo, en la imagen, la palabra del recuadro azul se llama palabra objetivo y las palabras de los recuadros blancos se llaman palabras de contexto en una ventana de tamaño 5.

### Source Text

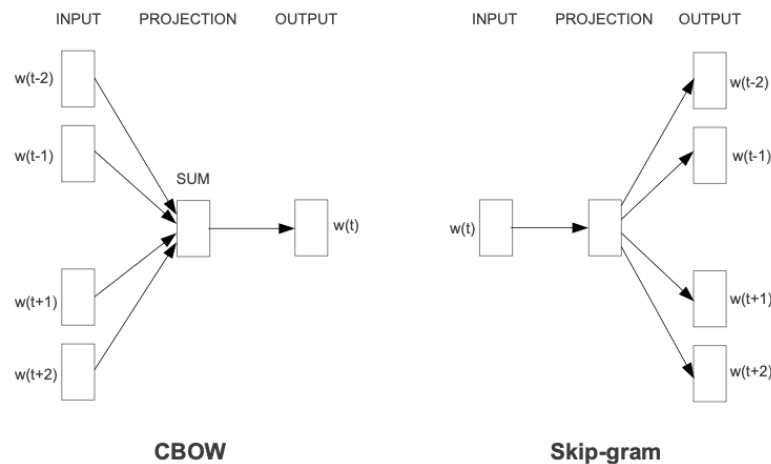
The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

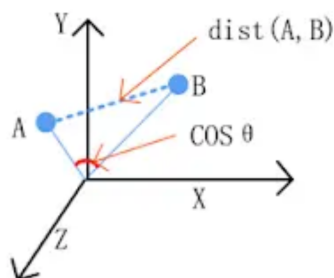
**CBOW** : El modelo se alimenta del contexto y predice la palabra objetivo. El resultado de la capa oculta es la nueva representación de la palabra ( $h_1, \dots, h_N$ ).

**Skip Gram**: El modelo se alimenta de la palabra objetivo y predice las palabras del contexto. El resultado de la capa oculta es la nueva representación de la palabra ( $h_1, \dots, h_N$ ).



Una vez teniendo el modelo entrenado, puede ser interesante comparar la distancia entre las palabras vamos a revisar las métricas de similitud.

**Cosine similarity** se utiliza medir la distancia cuando la norma de los vectores no tiene importancia. Esta métrica capta la similitud entre dos palabras. Cuanto más se acerque la cosine similarity a 1, más relacionadas estarán las dos palabras.



$$dist(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|}$$

$$dist(A, B) = \frac{\sum_{i=1}^N A_i B_i}{\sqrt{\sum_{i=1}^N A_i^2} \sqrt{\sum_{i=1}^N B_i^2}}$$

## Entrenamiento

Un modelo **Word2vec** puede ser entrenado con **softmax jerárquico y/o muestreo negativo**. Para aproximar la verosimilitud logarítmica condicional, un modelo busca maximizar, el **método softmax jerárquico** utiliza un **árbol de Huffman** para reducir el cálculo. El método de muestreo negativo, por otro lado, aborda el problema de maximización minimizando la probabilidad logarítmica de las instancias negativas muestreadas. Se sabe que el **softmax jerárquico** trabaja mejor para palabras infrecuentes, mientras que el muestreo negativo funciona mejor para palabras frecuentes y mejor con vectores de pocas dimensionales. Cuando el número de épocas aumenta, el **softmax jerárquico** deja de ser útil.

## Desarrollo de algoritmo

El dataset para entrenar y probar el algoritmo se obtuvo de Kaggle (<https://www.kaggle.com/datasets/CooperUnion/cardataset>).

Este dataset está compuesto por las siguientes columnas:

- Make – car marker
- Model – car model
- Year – year the car was manufactured
- Engine Fuel Type – engine fuel type
- Engine HP – engine horse power(HP) i.e the power an engine produces
- Engine Cylinders – engine cylinders
- Transmission Type – ☐rice☐isión type
- Driven\_Wheels – driven wheels
- Number of Doors – number of doors in car
- Market Category – category the car fits in
- Vehicle Size – Vehicle Style
- highway MPG – highway speed depending on the gallon of petrol or diesel in its tank
- city mpg – city speed
- Popularity – popularity of the car among twitter users where the data was extracted from
- MSRP – manufacturer suggested retail price

**Primero se carga los datos** y se realiza un análisis exploratorio para identificar insights en los datos. Así como identificar posibles datos relevantes en la decisión de compra de coche y a su vez para la comparación.

```
In [3]: df = pd.read_csv('data.csv')
df.head()
```

Out [3]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	391
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	391
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	391
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	391
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	391

Se tienen 11,914 elementos, realizara limpieza de datos para evitar o reducir sesgos en el resultado.

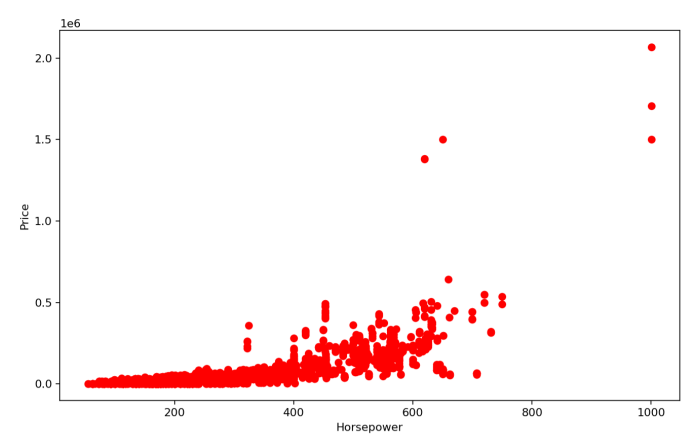
Matriz de correlación

Podemos ver como MSRP (precio) se correlaciona fuertemente con los caballos de fuerza, así como el número de cilindros.

Out [50]:

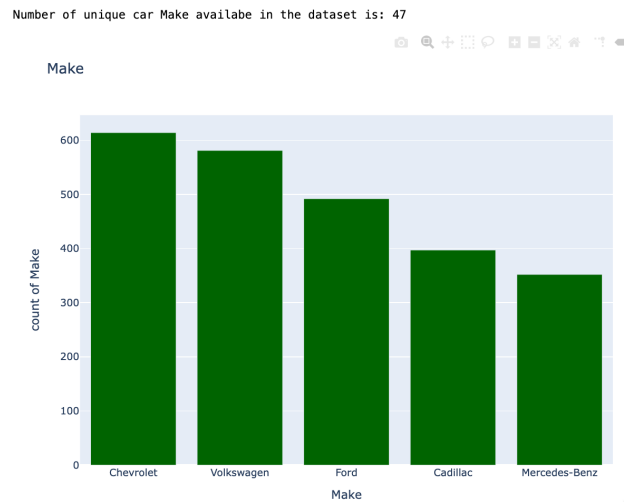
	Year	Engine HP	Engine Cylinders	Number of Doors	highway MPG	city mpg	Popularity	MSRP
Year	1.000000	0.351794	-0.041479	0.263787	0.258240	0.198171	0.073049	0.227590
Engine HP	0.351794	1.000000	0.779988	-0.102713	-0.406563	-0.439371	0.037501	0.662008
Engine Cylinders	-0.041479	0.779988	1.000000	-0.140088	-0.621606	-0.600776	0.041145	0.531312
Number of Doors	0.263787	-0.102713	-0.140088	1.000000	0.118570	0.120881	-0.048272	-0.126635
highway MPG	0.258240	-0.406563	-0.621606	0.118570	1.000000	0.886829	-0.020991	-0.160043
city mpg	0.198171	-0.439371	-0.600776	0.120881	0.886829	1.000000	-0.003217	-0.157676
Popularity	0.073049	0.037501	0.041145	-0.048272	-0.020991	-0.003217	1.000000	-0.048476
MSRP	0.227590	0.662008	0.531312	-0.126635	-0.160043	-0.157676	-0.048476	1.000000

En este grafico vemos como el precio se incrementa en función de los caballos de fuerza.

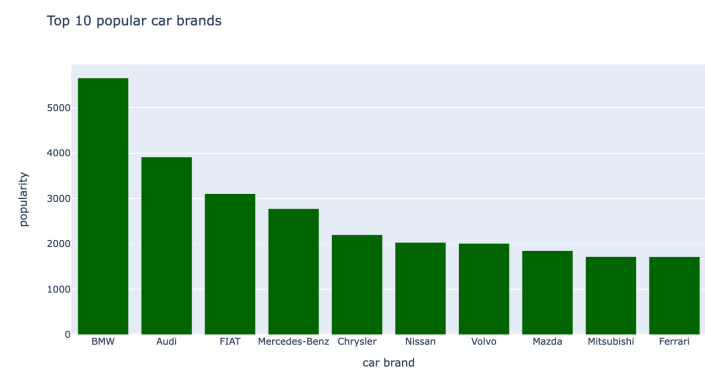


Es posible que nuestra decisión de seleccionar un coche también está ligada al precio, por lo que será importante poner atención a los caballos de fuerza y numero de cilindros. Por lo que podrá servir de punto de partida y pensar en cual es nuestro presupuesto.

Vemos las marcas con más modelos, aunque esto no necesariamente es muy relevante para el cliente promedio, pero es importante saber que tana diversidad de modelos se tener para una marca.



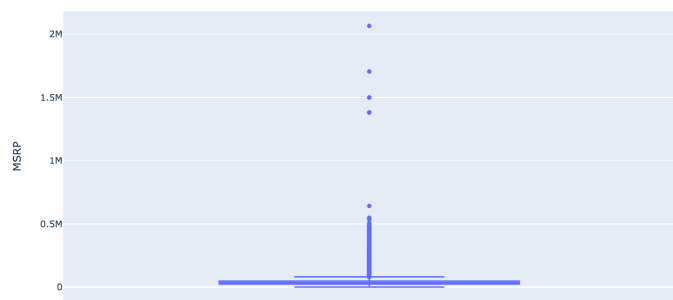
Quizá para algunas personas es relevante saber la marca más popular, pero regresamos al tema del precio y a su vez el presupuesto, sin embargo, como cliente nos gusta saber el nivel de popularidad de lo que compramos.



En este caso vemos como fluctúa el precio a razón del año del coche aquí es importante para el cliente saber que su coche podrá tener un buen valore de re-venta.



En este dataset vemos que los precios son muy parecidos con excepción de algunos modelos.



## Resultados

A continuación, se realizará la reestructura de los datos para usar word2Vec

```
Out[189]: [['premium unleaded (required)',
            'MANUAL',
            'rear wheel drive',
            'Factory Tuner',
            'Luxury',
            'High-Performance',
            'Compact',
            'Coupe',
            '46135',
            'BMW 1 Series M'],
            ['premium unleaded (required)',
            'MANUAL',
            'rear wheel drive',
            'Luxury',
            'Performance',
            'Compact',
            'Convertible',
            '40650',
            'BMW 1 Series'],
            ['premium unleaded (required)',
            'MANUAL',
            'rear wheel drive',
            'Luxury',
            'High-Performance',
            'Compact',
            'Coupe',
            '36350',
            'BMW 1 Series']]
```

Podemos observar que se concateno marca y modelo para poder identificarla de manera única más fácilmente.

Ahora procederemos con estos datos realizar el entrenamiento de la red neuronal con la librería Gensim- Word2Vec. Usaremos la estructura previamente generada, aquí será por medio de redes neuronales poco profundas para el procesamiento natural de lenguaje y así poder identificar similitudes entre las palabras que describen cada característica de los coches.

```
## entrenamiento word2vec
model = Word2Vec(sent, min_count=1, workers=4, vector_size = 50, window =4, sg = 1)
```

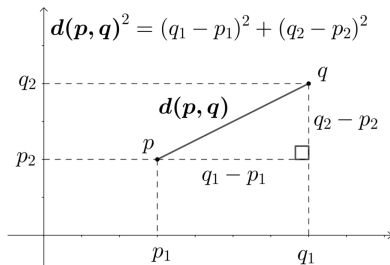
```
INFO - 10:47:18: collecting all words and their counts
INFO - 10:47:18: PROGRESS: at sentence #0, processed 0 words, keeping 0 word types
INFO - 10:47:18: collected 5412 word types from a corpus of 69369 raw words and 8084 sentences
INFO - 10:47:18: Creating a fresh vocabulary
INFO - 10:47:18: Word2Vec lifecycle event {'msg': 'effective_min_count=1 retains 5412 unique words (100.00% of original 5412, drops 0)', 'datetime': '2022-07-23T10:47:18.659488', 'gensim': '4.2.0', 'python': '3.8.8 (default, Apr 13 2021, 12:59:45) \n[Clang 10.0.0 ]', 'platform': 'macOS-10.16-x86_64-i386-64bit', 'event': 'prepare_vocab'}
INFO - 10:47:18: Word2Vec lifecycle event {'msg': 'effective_min_count=1 leaves 69369 word corpus (100.00% of original 69369, drops 0)', 'datetime': '2022-07-23T10:47:18.660190', 'gensim': '4.2.0', 'python': '3.8.8 (default, Apr 13 2021, 12:59:45) \n[Clang 10.0.0 ]', 'platform': 'macOS-10.16-x86_64-i386-64bit', 'event': 'prepare_vocab'}
INFO - 10:47:18: deleting the raw counts dictionary of 5412 items
INFO - 10:47:18: sample=0.001 downsamples 34 most-common words
INFO - 10:47:18: Word2Vec lifecycle event {'msg': 'downsampling leaves estimated 29165.091621406842 word corpus (42.00% of prior 69369)', 'datetime': '2022-07-23T10:47:18.689534', 'gensim': '4.2.0', 'python': '3.8.8 (default, Apr 13 2021, 12:59:45) \n[Clang 10.0.0 ]', 'platform': 'macOS-10.16-x86_64-i386-64bit', 'event': 'prepare_vocab'}
INFO - 10:47:18: estimated required memory for 5412 words and 50 dimensions: 4870800 bytes
INFO - 10:47:18: resetting layer weights
INFO - 10:47:18: Word2Vec lifecycle event {'update': False, 'trim_rule': 'None', 'datetime': '2022-07-23T10:47:18.743028', 'gensim': '4.2.0', 'python': '3.8.8 (default, Apr 13 2021, 12:59:45) \n[Clang 10.0.0 ]', 'platform': 'macOS-10.16-x86_64-i386-64bit', 'event': 'build_vocab'}
INFO - 10:47:18: Word2Vec lifecycle event {'msg': 'training model with 4 workers on 5412 vocabulary and 50 features, using sg=1 hs=0 sample=0.001 negative=5 window=4 shrink_windows=True', 'datetime': '2022-07-23T10:47:18.743537', 'gensim': '4.2.0', 'python': '3.8.8 (default, Apr 13 2021, 12:59:45) \n[Clang 10.0.0 ]', 'platform': 'macOS-10.16-x86_64-i386-64bit', 'event': 'train'}
INFO - 10:47:18: EPOCH 0: training on 69369 raw words (29201 effective words) took 0.0s, 1193402 effective words/s
INFO - 10:47:18: EPOCH 1: training on 69369 raw words (29175 effective words) took 0.0s, 1115114 effective words/s
INFO - 10:47:18: EPOCH 2: training on 69369 raw words (29196 effective words) took 0.0s, 1144031 effective words/s
INFO - 10:47:18: EPOCH 3: training on 69369 raw words (29274 effective words) took 0.0s, 1176877 effective words/s
INFO - 10:47:18: EPOCH 4: training on 69369 raw words (29192 effective words) took 0.0s, 1070981 effective words/s
INFO - 10:47:18: Word2Vec lifecycle event {'msg': 'training on 346845 raw words (146038 effective words) took 0.2s, 860741 effective words/s', 'datetime': '2022-07-23T10:47:18.913561', 'gensim': '4.2.0', 'python': '3.8.8 (default, Apr 13 2021, 12:59:45) \n[Clang 10.0.0 ]', 'platform': 'macOS-10.16-x86_64-i386-64bit', 'event': 'train'}
INFO - 10:47:18: Word2Vec lifecycle event {'params': 'Word2Vecvocabulary=5412, vector_size=50, alpha=0.025', 'datetime': '2022-07-23T10:47:18.913902', 'gensim': '4.2.0', 'python': '3.8.8 (default, Apr 13 2021, 12:59:45) \n[Clang 10.0.0 ]', 'platform': 'macOS-10.16-x86_64-i386-64bit', 'event': 'created'}
```

Un punto importante es ver un parámetro donde se indica el modelo a usar CBOW / Skin Gram:

`model = Word2Vec(sent, min_count=1, workers=4, vector_size = 50, window =4, sg = 1)`

**sg:** El algoritmo de entrenamiento, ya sea CBOW(0) o skip gram (1).

Por default este algoritmo utiliza la distancia euclidiana, la cual se basa en el teorema de Pitágoras, según el cual, la hipotenusa al cuadrado es igual a la suma de catetos al cuadrado.



Observemos los resultados, en el caso del coche Mazda 3, podemos ver como quedo posicionado en el modelo:

```
In [191]: # incrustación del modelo de entrenamiento
model.wv['Mazda 3']

Out[191]: array([ 0.06533656,  0.3055265 , -0.27457634,  0.13298537,  0.04991212,
 -0.4050116 , -0.05869852,  0.18770173,  0.02597883, -0.38327074,
  0.2517024 ,  0.0347651 ,  0.04122006,  0.0109099 , -0.2226513 ,
  0.54806656,  0.4973844 , -0.03578791, -0.58504057, -0.5539398 ,
  0.40824774,  0.32722402,  0.4448862 , -0.02459279,  0.204835 ,
  0.43684605,  0.09780576,  0.26310462, -0.21231827,  0.3502291 ,
  0.08193156, -0.11249212,  0.00084663, -0.09684087, -0.4326018 ,
 -0.19292152,  0.21028283,  0.37262213,  0.19632646, -0.01213402,
  0.12170535,  0.03895576, -0.01359789,  0.23864971,  0.5389929 ,
  0.16559936, -0.04918798,  0.00771399,  0.18210077,  0.2222505 ],
 dtype=float32)
```

## Comparacion entre modelos

Calculamos la similitud entre dos modelos de autos. El resultado de ese calculo ser la similitud euclidiana entre dos modelos.

```
In [169]: model.wv.similarity('Chrysler 300','Mazda 3')

Out[169]: 0.7761583

In [170]: model.wv.similarity('Honda Civic','Mazda 3')

Out[170]: 0.92887914

In [171]: model.wv.similarity('Mazda CX-5', 'Hyundai Tucson')

Out[171]: 0.96451104
```

También podemos ver como al hacer comparaciones entre coches que por sentido común sabemos que son similares los resultados arrojados dan resultados muy satisfactorios.

Por ejempló, es normal que el Chrysler 300, no se tan parecido al Mazda 3 debido a que Chrysler 300 tiene un motor más grande y tiene más lujos en el interior y con ellos un precio de más de 300,000 de diferencia. Así como también es normal que un Honda Civic y un; Mazda 3 se parezcan arriba del 90% debido a que son del mismo segmento con precios muy similares.



## Obtener los mas similares a un determinado modelo

```
In [172]: model.wv.most_similar('Mazda 3',topn=15)
```

```
Out[172]: [('Chevrolet Sonic', 0.9917768239974976),
('Toyota Yaris', 0.9895885586738586),
('Volkswagen Beetle', 0.9887951612472534),
('Honda CR-Z', 0.9887909889221191),
('Honda Fit', 0.9876793622970581),
('Mitsubishi Expo', 0.9876027703285217),
('Mitsubishi Eclipse', 0.9874030351638794),
('Chevrolet Spark', 0.9852787256240845),
('Suzuki SX4', 0.9845237731933594),
('Volkswagen Rabbit', 0.9840819239616394),
('Volkswagen Golf', 0.9839626550674438),
('Nissan Versa Note', 0.982747495174408),
('Nissan Juke', 0.9826412200927734),
('Toyota Prius', 0.9819141626358032),
('Ford Aspire', 0.9815130233764648)]
```

```
In [173]: model.wv.most_similar('Honda Civic')
```

```
Out[173]: [('Acura Integra', 0.9852666258811951),
('Pontiac Firebird', 0.9842002987861633),
('Volkswagen GLI', 0.984021008014679),
('Audi 90', 0.9807649254798889),
('Volvo 240', 0.9790794253349304),
('Acura ILX', 0.9768791198730469),
('30995', 0.9765002727508545),
('32550', 0.9753777980804443),
('Acura Legend', 0.9753730893135071),
('Volkswagen Golf R', 0.9749211072921753)]
```

```
In [174]: model.wv.most_similar('Mazda CX-5',topn=15)
```

```
Out[174]: [('Nissan Murano', 0.9959679841995239),
('Toyota RAV4', 0.9938620328903198),
('Toyota Highlander', 0.993678629398346),
('Honda Element', 0.9933426380157471),
('Buick Encore', 0.993011653423093),
('Chevrolet Traverse', 0.9927770495414734),
('Nissan Rogue', 0.9924910664558411),
('Dodge Journey', 0.9919325709342957),
('Honda Pilot', 0.9918960332870483),
('Cadillac SRX', 0.9906962513923645),
('Hyundai Santa Fe', 0.9904412627220154),
('Buick Enclave', 0.9903085231781006),
('Mazda CX-7', 0.9902036190032959),
('Suzuki XL7', 0.9894177913665771),
('Subaru Forester', 0.9892865419387817)]
```

Usando sg = 0

```
## entrenamiento word2vec
model = Word2Vec(sent, min_count=1,workers=4, vector_size = 50, window =4, sg = 0)
```

Como podemos ver en los resultados con sg=0 se incrementa la similitud con algunos modelos

```
In [192]: model.wv['Mazda 3']
Out[193]: array([ 0.05263817,  0.21799384,  0.07126995,  0.13245273,  0.14845437,
 -0.14425816,  0.10813771,  0.28085655, -0.1444299, -0.13223447,
 -0.00144379, -0.13119578,  0.0709377, -0.07073572, -0.11359424,
  0.28588614,  0.2304885,  0.10544287, -0.26508775, -0.40329778,
  0.09257998,  0.15477091,  0.4410756, -0.12236764,  0.06150801,
  0.13024686,  0.01449828,  0.05444625, -0.08107545, -0.00675763,
  0.02145168, -0.10652399,  0.07459471, -0.06441473, -0.10944897,
 -0.1453692,  0.21519105,  0.10436839, -0.02081411,  0.01423582,
  0.16896032,  0.0168284, -0.01536861,  0.09467502,  0.36891985,
  0.12568736,  0.05851281, -0.1214141,  0.38242728, -0.03637101],
 dtype=float32)
```

## Comparacion entre modelos

Calculamos la similitud entre dos modelos de autos. El resultado de ese calculo ser la similitud euclidiana entre dos modelos.

```
In [194]: model.wv.similarity('Chrysler 300','Mazda 3')
```

```
Out[194]: 0.96146655
```

```
In [195]: model.wv.similarity('Honda Civic','Mazda 3')
```

```
Out[195]: 0.98456466
```

```
In [196]: model.wv.similarity('Mazda CX-5', 'Hyundai Tucson')
```

```
Out[196]: 0.97571886
```

También podemos ver que cambia la similitud con algunos coches, basado en la experiencia aparentemente es más aceptable este último anqué los resultados no son muy diferentes.

### Obtener los mas similares a un determinado modelo

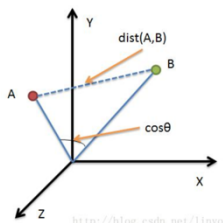
```
In [197]: model.wv.most_similar('Mazda 3',topn=15)
Out[197]: [('Honda Fit', 0.9944913387298584),
('Volkswagen New Beetle', 0.9940776228904724),
('electric', 0.993954062461853),
('DIRECT_DRIVE', 0.9937862753868103),
('Toyota Celica', 0.9935608506202698),
('Toyota Yaris', 0.9927308559417725),
('Subaru Impreza', 0.9925972819328308),
('Nissan Versa Note', 0.9922395944595337),
('Volkswagen Golf', 0.9916266202926636),
('Acura Integra', 0.9907575845718384),
('Volkswagen Rabbit', 0.9901294112205505),
('Chevrolet Malibu Maxx', 0.9894833564758301),
('Honda Accord Crosstour', 0.989478349685669),
('Honda Insight', 0.9892844557762146),
('Hyundai Accent', 0.9891670942306519)]

In [198]: model.wv.most_similar('Honda Civic')
Out[198]: [('Audi A3', 0.9947556257247925),
('Volkswagen CC', 0.9946312308311462),
('Volkswagen Jetta GLI', 0.9942192435264587),
('Saab 9-3', 0.9940037131309509),
('Ford Focus', 0.992764115335571),
('Acura Integra', 0.9913730025291443),
('DIRECT_DRIVE', 0.9905415177345276),
('Pontiac Firebird', 0.9905280470848083),
('Honda Accord', 0.990498960818158),
('Audi A4', 0.9903438091278076)]

In [199]: model.wv.most_similar('Mazda CX-5',topn=15)
Out[199]: [('Dodge Journey', 0.9942284822463989),
('Suzuki XL7', 0.9926245212554932),
('Honda Pilot', 0.9924271106719971),
('Nissan Murano', 0.9922621846199036),
('GMC Acadia', 0.9922279715538025),
('Cadillac SRX', 0.9910781979560852),
('Subaru Forester', 0.9906954169273376),
('Chevrolet Equinox', 0.9905372262001038),
('Nissan Pathfinder', 0.9901095032691956),
('Mitsubishi Outlander', 0.9900754690170288),
('Mazda Tribute', 0.9894603490829468),
('Volkswagen Tiguan', 0.9892464876174927),
('Chevrolet Traverse', 0.9891505241394043),
('Nissan Rogue', 0.9891397953033447),
('Lincoln MKX', 0.9889180064201355)]
```

Como sabemos la distancia euclidiana pudiera no funcionar bien para los vectores de palabras de alta dimensión. Esto se debe a que la similitud euclidiana aumentará el número de dimensiones incluso si la palabra incrustación tiene diferentes significados. Por lo que aplicaremos la similitud del coseno para medir la similitud entre dos vectores. Matemáticamente, es el coseno del ángulo entre dos vectores proyectados en un espacio multidimensional. Por lo tanto, la similitud del coseno es el ángulo de los vectores de palabras y no la magnitud. Bajo la similitud del coseno, ninguna similitud se expresa como un ángulo de 90 grados, mientras que la similitud total de 1 está en un ángulo de 0 grados.

La fórmula de la similitud del coseno se calcula con la siguiente fórmula:

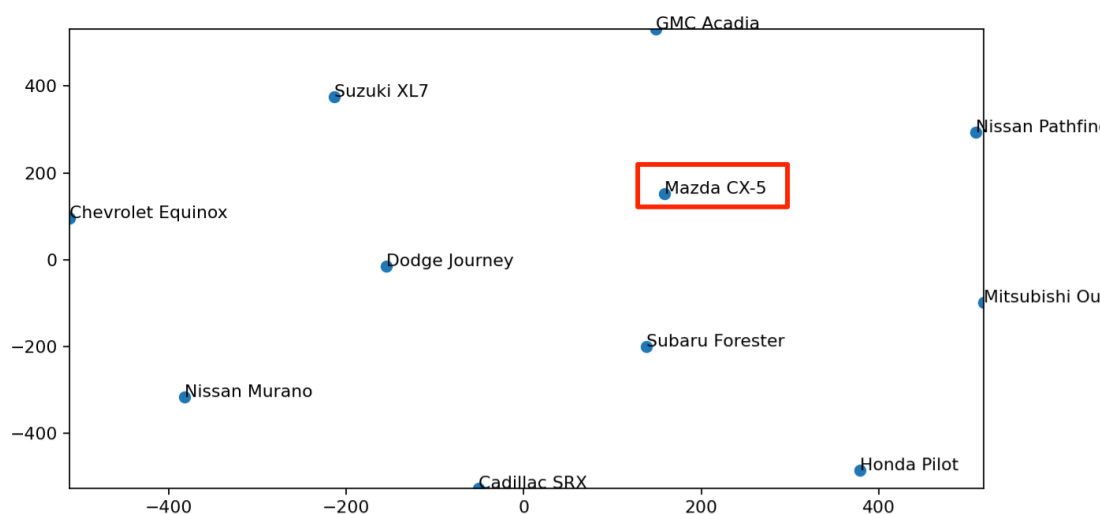
$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$


```
In [219]: ## mostrar los mas similares entre con el modelo en cuestion por similitud por coeseno
cosine_distance (model,'Mazda CX-5',Maker_Model,10)
Out[219]: [('Dodge Journey', 0.99422854),
('Suzuki XL7', 0.9926245),
('Honda Pilot', 0.9924271),
('Nissan Murano', 0.9922622),
('GMC Acadia', 0.99222785),
('Cadillac SRX', 0.9910782),
('Subaru Forester', 0.9906954),
('Chevrolet Equinox', 0.9905372),
('Nissan Pathfinder', 0.99010944),
('Mitsubishi Outlander', 0.9900755)]

In [218]: ## mostrar los mas similares entre con el modelo en cuestion
cosine_distance (model,'Honda Civic',Maker_Model,10)
Out[218]: [('Audi A3', 0.99475557),
('Volkswagen CC', 0.99463135),
('Volkswagen Jetta GLI', 0.99421936),
('Saab 9-3', 0.99400383),
('Ford Focus', 0.9927642),
('Acura Integra', 0.99137294),
('Pontiac Firebird', 0.99052805),
('Honda Accord', 0.99049896),
('Audi A4', 0.9903436),
('Volkswagen Beetle Convertible', 0.9902733)]
```

Podemos observar que los resultados usando distancia euclidiana y similitud por coseno son similares, por lo que seremos libres de usar cualquier de las dos en base a estas pruebas.

Aplicando T-SNE (t-distributed Stochastic Neighbor Embedding) para visualizar datos de alta dimensión al reducir el espacio dimensional mientras se mantiene la distancia relativa por pares entre los puntos. T-SNE representa los datos conservando las relaciones de vecindad.



Como podemos observar por este método es posible distinguir los coches más similares, y si lo analizamos detenidamente entre las marcas y modelos de coche coincide con el sentido común el parecido entre esos coches cercanos a la Mazda CX-5 y es normal tener cercanas Suv's

## Discusión

Es posible observar la efectividad del procesamiento mediante Word2Vec dado que coincide la similitud de determinada en función de la revisión de características y experiencia.

Cabe mencionar que este trabajo no está basado en un artículo científico si no más bien en una necesidad de un cliente que busca hacer una prueba de concepto con la compañía con la que en este momento laboro. Pero si se utilizo como base la librería Genism donde se está usando la metodología Word2Vec utilizando redes neuronales a mediane procesamiento natural de lenguaje. Así mismo se describió la metodología en relación con la base teoría de Word2Vec a modo de resumen sin entrar en mucho detalle, pero para dar un contexto de funcionamiento.

Posteriormente podemos observar en los resultados que son muy aceptables por que el parecido entre estos coches se puede apreciar desde la perspectiva de sentido común:

### Obtener los mas similares a un determinado modelo

```
model.wv.most_similar('Mazda 3',topn=15)
```

```
[('Honda Fit', 0.9944913387298584),
 ('Volkswagen New Beetle', 0.9940776228904724),
 ('electric', 0.993954062461853),
 ('DIRECT_DRIVE', 0.9937862753868103),
 ('Toyota Celica', 0.9935608506202698),
 ('Toyota Yaris', 0.9927308559417725),
 ('Subaru Impreza', 0.9925972819328308),
 ('Nissan Versa Note', 0.9922395944595337),
 ('Volkswagen Golf', 0.9916266202926636),
 ('Acura Integra', 0.9907575845718384),
 ('Volkswagen Rabbit', 0.9901294112205505),
 ('Chevrolet Malibu Maxx', 0.9894833564758301),
 ('Honda Accord Crosstour', 0.989478349685669),
 ('Honda Insight', 0.9892844557762146),
 ('Hyundai Accent', 0.9891670942306519)]
```

```
model.wv.most_similar('Honda Civic')
```

```
[('Audi A3', 0.9947556257247925),
 ('Volkswagen CC', 0.9946312308311462),
 ('Volkswagen Jetta GLI', 0.9942192435264587),
 ('Saab 9-3', 0.9940037131309509),
 ('Ford Focus', 0.9927641153335571),
 ('Acura Integra', 0.9913730025291443),
 ('DIRECT_DRIVE', 0.9905415177345276),
 ('Pontiac Firebird', 0.9905280470848083),
 ('Honda Accord', 0.990498960018158),
 ('Audi A4', 0.9903438091278076)]
```

```
model.wv.most_similar('Mazda CX-5',topn=15)
```

```
[('Dodge Journey', 0.9942284822463989),
 ('Suzuki XL7', 0.9926245212554932),
 ('Honda Pilot', 0.9924271106719971),
 ('Nissan Murano', 0.9922621846199036),
 ('GMC Acadia', 0.9922279715538025),
 ('Cadillac SRX', 0.9910781979560852),
 ('Subaru Forester', 0.9906954169273376),
 ('Chevrolet Equinox', 0.9905372262001038),
 ('Nissan Pathfinder', 0.9901095032691956),
 ('Mitsubishi Outlander', 0.9900754690170288),
 ('Mazda Tribute', 0.9894603490829468),
 ('Volkswagen Tiguan', 0.9892464876174927),
 ('Chevrolet Traverse', 0.9891505241394043),
 ('Nissan Rogue', 0.9891397953033447),
 ('Lincoln MKX', 0.9889180064201355)]
```

## Conclusiones y trabajo a futuro

Como vimos en la explicación y resultados podemos concluir que el algoritmo cumple con el objetivo a partir del dataset usado con las características seleccionadas, es importante mencionar que si agregamos más características al dataset sería posible tener más detalles que permitan la identificación de la similitud más adecuada como si estuviéramos viendo los diferentes folletos de características de los coches.

Un posible trabajo futuro será la evolución de este algoritmo usando más características para que se pueda valorar la similitud por ejemplo si un coche tiene faros de niebla, numero de bolsas de aire, tipos de frenos, tamaño de tanque de gasolina, etc. Para esto se tendría que buscar otro set de datos o complementarlo para poder hacer un procesamiento mas personalizado a la información que al día de hoy se muestra en los folletos de los coches.

## Referencias

- Goldberg, Y. (2014, February 15). *word2vec Explained: deriving Mikolov et al.'s*. . . arXiv.Org. <https://arxiv.org/abs/1402.3722>
- Adewumi, T. P. (2020, March 23). *Word2Vec: Optimal Hyper-Parameters and Their Impact on NLP Downstream Tasks*. arXiv.Org. <https://arxiv.org/abs/2003.11645>
- Cerisara, C. (2020, October 22). *On the Effects of Using word2vec Representations in Neural*. . . arXiv.Org. <https://arxiv.org/abs/2010.11490>
- Vijayakumar, A. K. (2017, March 6). *Sound-Word2Vec: Learning Word Representations Grounded in Sounds*. arXiv.Org. <https://arxiv.org/abs/1703.01720>
- Rong, X. (2014, November 11). *word2vec Parameter Learning Explained*. arXiv.Org. <https://arxiv.org/abs/1411.2738>
- Tomas Mikolov (2013). <https://www.researchgate.net/> . [https://www.researchgate.net/publication/257882504\\_Distributed\\_Representations\\_of\\_Words\\_and\\_Phrases\\_and\\_their\\_Compositionality](https://www.researchgate.net/publication/257882504_Distributed_Representations_of_Words_and_Phrases_and_their_Compositionality)