Introducción a la Ciencia de Datos

Tarea 2

2024

Leandro Cantera, Miguel Paolino

Parte 1

1. División de los párrafos de entrenamiento y test

A partir de los datos "limpios" de la tarea 1, armamos un dataset que contiene el número de párrafo, el texto limpio, el nombre del personaje, el título y el genero de la obra, para tres personajes (Antony, Cleopatra, Queen Margaret), llamado df_dataset.

Se divide el dataset en un conjunto de entrenamiento del 70% de los datos, y otro de testeo del el 30% de los datos, usando la función *train_test_split* de *sklearn.model_selection*. Se utiliza el muestreo estratificado y se fija el parámetro *random state* para hacer reproducible los resultados.

Fuente

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=123, stratify=y)
```

Los tamaños de Train y Test son 438 y 188 párrafos, respectivamente.

2. Visualización del balance de párrafos entre los personajes

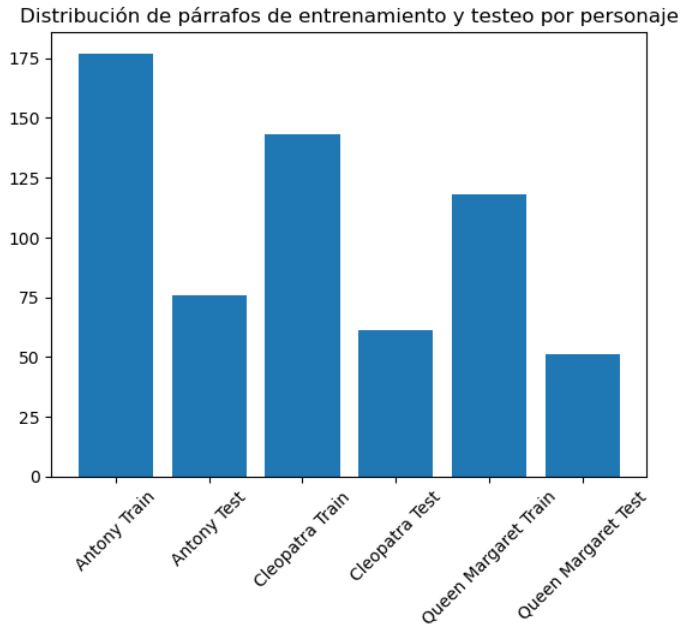

Figura 1. Distribución de párrafos en los conjuntos de entrenamiento (train) y testeo (test) de cada uno de los personajes.

3. Bag of Words (BoW)

La técnica de la Bag of Words se usa para el análisis del lenguaje natural, La técnica consiste en representar un documento de texto como un conjunto o bolsa de palabras, desestimando el orden y la estructura gramatical de las palabras en el texto.

Mediante esta técnica, se crea una matriz donde los valores de las columnas representan la cantidad de veces que aparece una palabra del vocabulario (corpus) en un texto. Los textos son representados por las filas. La matriz resultante es generalmente una *sparse matrix*, ya que en la mayoría de los casos el valor va a ser cero (la palabra del vocabulario -columna- no está presente en el texto -fila-). En este caso, como estamos trabajando sobre el subconjunto de entrenamiento de 438 párrafos, es una matriz de 438 filas por 2776 columnas (cantidad de palabras) y es dispersa porque usa una cantidad reducida de palabras del vocabulario en cada párrafo.

4. N-gram y Term Frequency-Inverse Document Frequency

Como el BoW no contempla el orden de las palabras, en principio no podría distinguir si hay ciertas agrupaciones de palabras (como los *phrasal verbs*) y consecuentemente podría perder algo de contexto. Para esto se puede ampliar el rango de n-grams para devolver bigrams (2-grams) o trigrams (3-grams), que serían agrupaciones de 2 o 3 palabras, respectivamente, ayudando a dar algo de contexto a cada palabra.

Por otro lado, el tf-idf pondera la frecuencia de la palabra en el total del texto, según las siguientes ecuaciones:

- $tf = \frac{count}{total}$, donde count es el número de ocurrencias de la palabra y total es el número de palabras en el texto.
- $idf = log\frac{N}{n}$ con N el número de instancias del conjunto y n el número de instancias que la palabra aparece en el texto.
- $tf.idf = td \times idf$

Fuente

```
tf_idf = TfidfTransformer(use_idf=False)
X_train_tf = tf_idf.fit_transform(X_train_counts)
X_train_tf
```

5. PCA para para las 2 primeras componentes

La varianza explicada del PCA para los 2 componentes principales es 7% y la visualización es:

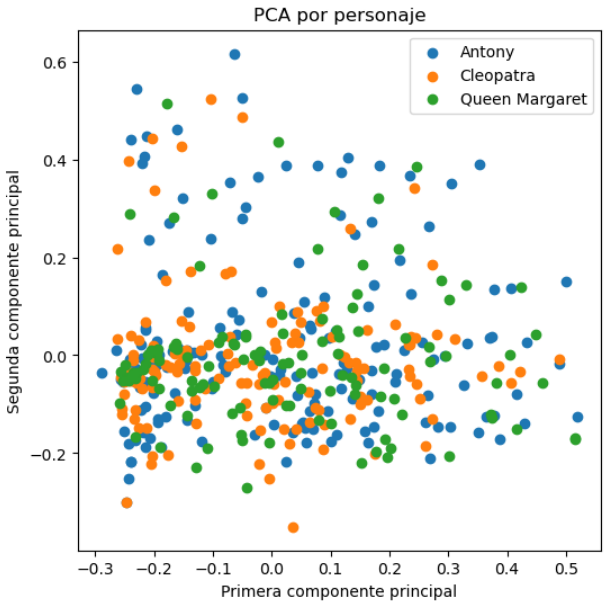

Figura 2. PCA para conjunto de entrenamiento, con stopwords = None, ngram = (1,1) e idf = True.

Si filtramos por las stop words del idioma inglés, el parámetro `use_idf=True` y `ngram_range=(1,2)` obtenemos una varianza explicada de 1,6%

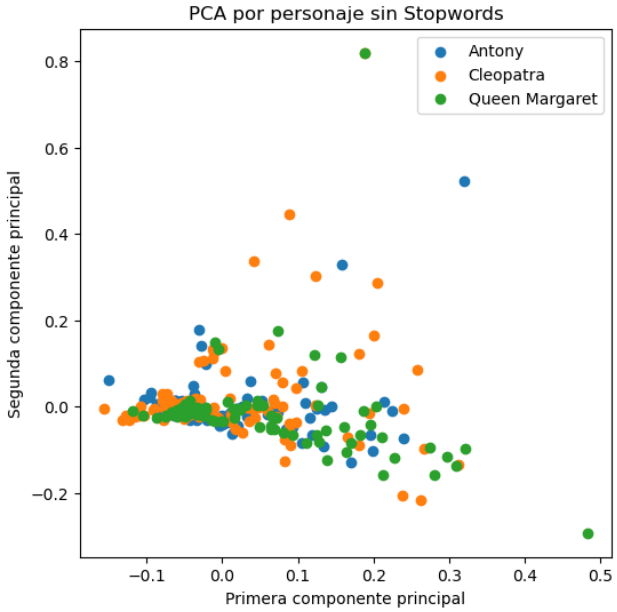

Figura 3. PCA para conjunto de entrenamiento, con stopwords = "english", ngram = (1,2) e idf = True.

En ambos casos, es posible afirmar que solamente 2 componentes principales no son suficientes ya que, en ambos casos, la varianza explicada es muy baja. Las figuras a continuación nos muestran la varianza explicada acumulada a medida que aumentan los componentes principales. Se puede observar que con 100 componentes principales se llega a explicar el 70% de la varianza.

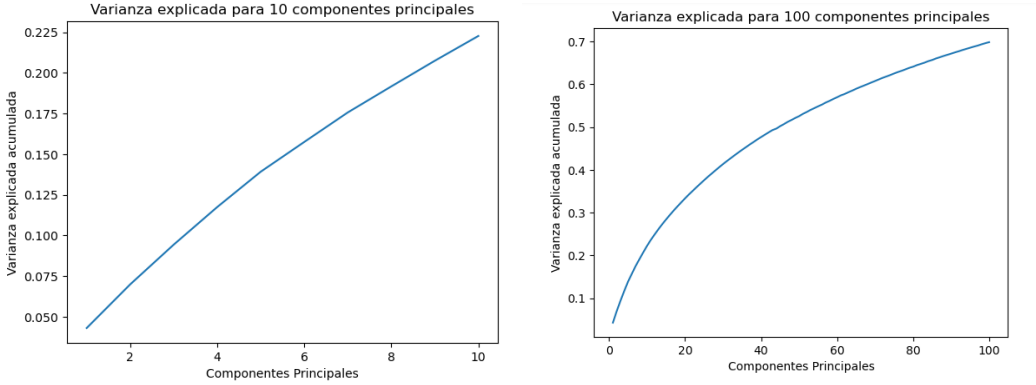

Figura 4. Varianza explicada acumulada para el caso de stopwords = "english" y idf = (1,2). 10 componentes principales (izquierda) y 100 componentes principales (derecha)

Parte 2

1. Entrenamiento de modelo con Multinomial Naive Bayes

Definimos los siguientes indicadores para evaluar el comportamiento del modelo seleccionado:

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN}$$

$$Precision = \frac{VP}{VP + FP}$$

$$Recall = \frac{VP}{VP + FN}$$

Donde:

VP = Verdaderos Positivos
VN = Verdaderos Negativos
FP = Falsos Positivos
FN = Falsos Negativos

Los indicadores para el modelo *Multinomial Naive Bayes* para los subsets de entrenamiento y testeo son:

Tabla 1. Comparación de accuracy, precisión y recall del modelo Multinomial Naive Bayes para los subconjuntos de entrenamiento (Train) y testeo (Test). Tomando como parámetros: stopwords = None, ngram = (1,1), idf = True

	Accuracy	**Precision**	**Recall**
Train	0.66	0.82	0.62
Test	0.48	0.72	0.42

La matriz de confusión para el conjunto de testeo se muestra en la siguiente figura:

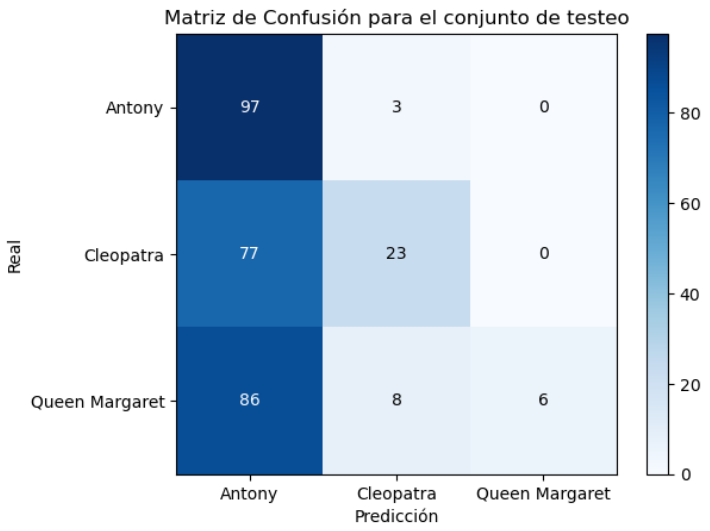

Figura 5. Matriz de confusión para el conjunto de testeo obtenida usando el modelo Multinomial Naive Bayes, con parámetros stopwords = None, ngram = (1,1), idf = True.

El valor de *accuracy* es un parámetro útil para evaluar los modelos de clasificación, cuando hay desbalance entre las clases del dataset (como se ve en la matriz de confusión) es necesario usar también la *precision* y *recall*, para conocer si nuestro modelo es efectivo aislando los falsos positivos.

En el caso de análisis, los bajos valores de *accuracy* y *recall* nos indican que es bueno prediciendo los casos positivos pero es conservador, sobretodo porque el valor de *precision* es más alto.

2. Cross-validation

Se usa la cross-validation para no separar datos para la validación que no se puedan usar para el entrenamiento, y así no reducir la cantidad de datos usados para entrenar el modelo. Además, ayuda a disminuir la varianza de los resultados al obtenerlo como el promedio de varias evaluaciones.

Dividimos el dataset en 4 partes y usamos 3 para entrenar y 1 para validar. Los parámetros para comparar son:

- Usar stop words o no.
- El tamaño del n-grama.
- Si se usa idf o no.

Se hace un gráfico de violín (Figura 6) con los resultados de *accuracy* para los diferentes sets de parámetros.

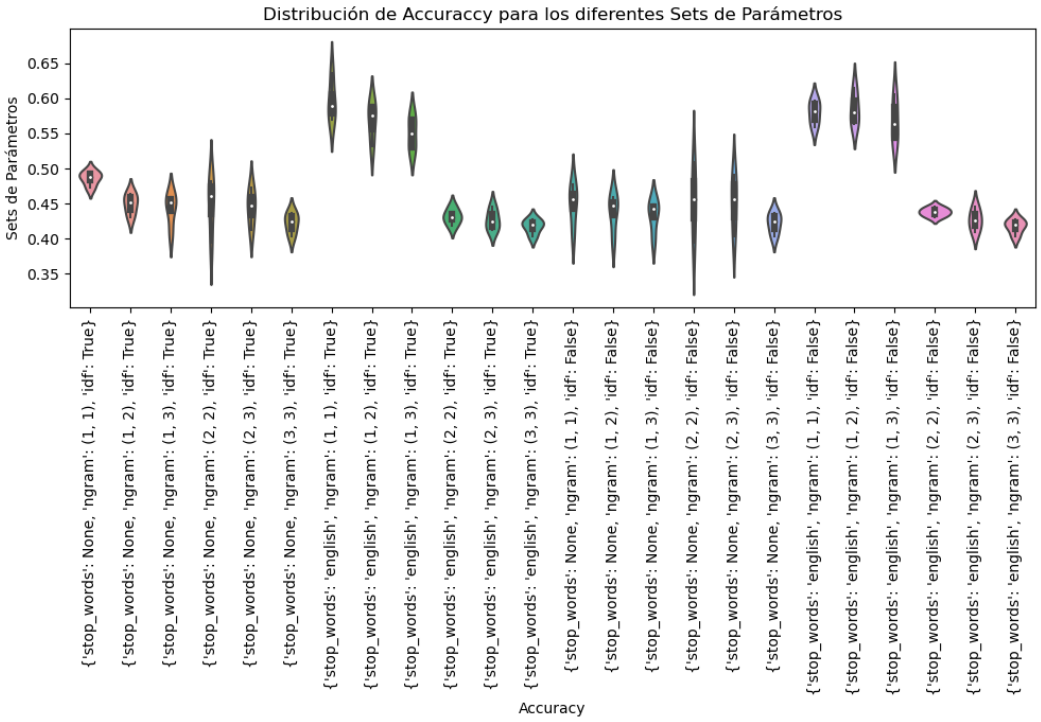

Figura 6. Grafico de violín con la distribución de los datos de accuracy obtenidos de la validación cruzada de 4 subsets de train/validation con los diferentes parámetros de stopwords, ngram e idf.

3. Mejor modelo

El mejor modelo fue con los parámetros stopwords = “english”, ngram = (1,1) e idf = False

Los resultados obtenidos son:

Tabla 2. Comparación de accuracy, precisión y recall del modelo Multinomial Naive Bayes para los subconjuntos de entrenamiento (Train) y testeo (Test). Tomando como parámetros: stopwords = “english”, ngram = (1,1), idf = False.

	Accuracy	**Precision**	**Recall**
Train	0.808	0.880	0.759
Test	0.606	0.678	0.577

La matriz de Confusión para estos parámetros es:

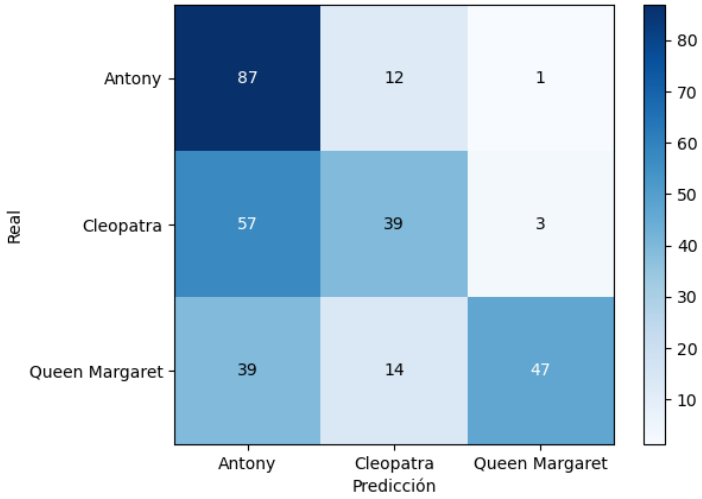

Figura 7. Matriz de confusión para el conjunto de testeo obtenida usando el modelo Multinomial Naive Bayes, con parámetros stopwords = "english", ngram = (1,1), idf = False.

Las limitaciones del BoW vienen por:

- La pérdida del orden de las palabras, su estructura y semántica.
- Si el vocabulario es muy grande genera matrices dispersas muy grandes.

Las limitaciones del TF-IDF vienen por:

- No tiene análisis contextual de como se usan las palabras en una oración.
- Si hay palabras en el subconjunto test que no fueron usadas en train, el modelo las ignora.
- No contempla el orden de las palabras .

4. Evaluación del modelo Support Vector Classification (SVC)

Las Support Vector Machines (SVM) son un conjunto de métodos de aprendizaje supervisado utilizados para la clasificación, la regresión y la detección de outliers. El objetivo del algoritmo SVM es encontrar un hiperplano que separe de la mejor forma posible dos clases diferentes de puntos de datos. Para esto define el margen como la distancia entre los puntos más cercanos. El objetivo es seleccionar un hiperplano con el máximo margen posible entre vectores de soporte en el conjunto de datos dado.

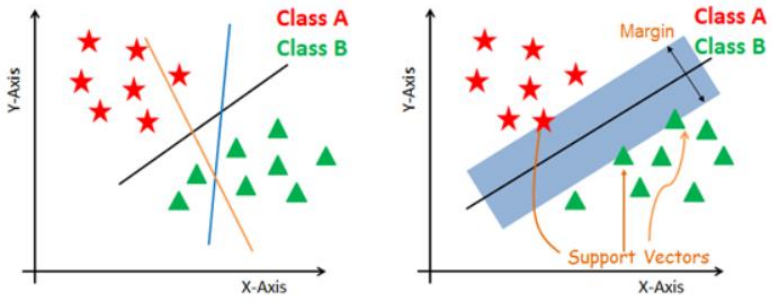

Figura 8. Representación gráfica bidimensional del SVM.

Dentro de este conjunto de métodos, en *scikit-learn* está el Support Vector Classification (SVC).

Se aplicó el SVC para los mismos parámetros para los cuales el Multinomial Naive Bayes había obtenido mejor accuracy (stopwords = "english", ngram = (1,1) e idf = False). Si comparamos los resultados obtenidos con el Multinomial Naive Bayes con los resultados obtenidos al aplicar el método de Support Vector Classification, podemos ver que el SVC tiene una mucho mejor accuracy para el conjunto de entrenamiento, pero relativamente similar desempeño para el conjunto de testeo. En base a esto podríamos inferir que el SVC está sobreajustado, teniendo muy buen rendimiento cuando se lo corre sobre los datos que entrenó, pero un desempeño bastante menor cuando se usa el subconjunto de testeo para validar el clasificador.

Tabla 3. Comparación de valores de accuracy para los modelos Multinomial Naive Bayes y Support Vector Classification, tanto para el subconjunto de entrenamiento (Train) como el subconjunto de testeo (Test).

	MultinomialNB	**SVC**
Train	0.81	0.98
Test	0.61	0.57

Al comparar las matrices de confusión sobre el subconjunto de testeo del Multinomial Naive Bayes (Figura 7) y el SVC (Figura 9), se evidencia como los valores de *accuracy* son efectivamente similares. El MultinomialNB obtuvo valores predichos correctos en un 87%, 39% y 47% para Antony, Cleopatra y Queen Margarte, respectivamente. Por otro lado, el SVC obtuvo valores predichos correctos en un 76%, 38% y 51% para los mismos tres personajes, respectivamente.

De todas formas, es posible que algún set de parámetros diferente pudiera dar mejores resultados para el SVC ya que no se chequeó el mejor set de hiper-parámetros mediante validación cruzada para ese modelo.

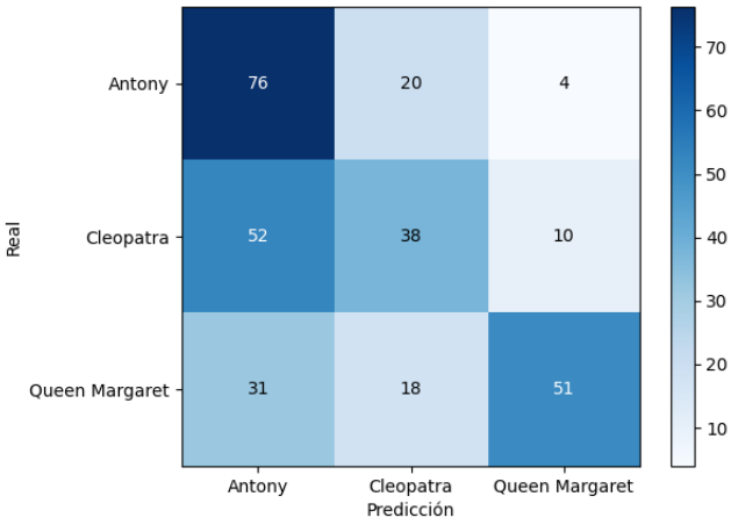

Figura 9. Matriz de confusión para el conjunto de testeo obtenida usando el modelo Supported Vector Classification, con parámetros stopwords = "english", ngram = (1,1), idf = False.

5. Cambio de personaje

Se elije cambiar al personaje con menores apariciones (Queen Margaret) por un personaje con más apariciones que Antony. Se decide trabajar con Henry V, el personaje que tiene mayor cantidad de palabras asignadas en toda la base de datos (luego de *Poet* y *stage directions*). Al volver a ejecutar el modelo Multinomial Naive Bayes, se puede ver como el sesgo que, en la matriz de confusión de la figura 7 hay hacia Antony (el modelo predice erróneamente que muchos de los párrafos son dichos por Antony), ahora está hacia Henry V (Figura 10).

Para evitar este sesgo, sería de utilidad generar un conjunto de datos en los que todos los personajes estudiados tengan la misma cantidad de intervenciones, y un número similar de palabras asignadas a ellos.

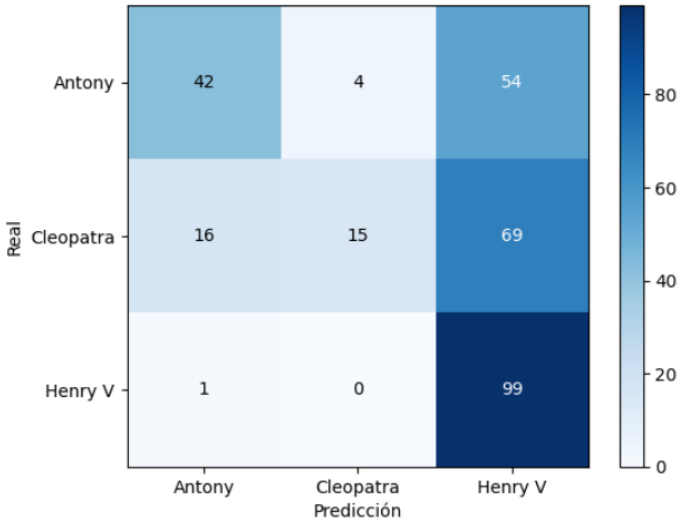

Figura 10. Matriz de confusión para el conjunto de testeo obtenida usando el modelo Multinomial Naive Bayes, con parámetros stopwords = "english", ngram = (1,1), idf = False. En este caso para los personajes Antony, Cleopatra y Henry V.

6. Técnica alternativa

De acuerdo a las limitaciones que presentan los resultados de BoW y TF-IDF vistos en el punto (3), se exploró otra técnica para el procesamiento de lenguaje natural, la recomendada es *Word Embeddings.*

Los *word embeddings* son representaciones vectoriales densas de palabras. En lugar de representar las palabras como vectores dispersos codificados en caliente (donde cada palabra es una dimensión separada), los *word embeddings* mapean las palabras a un espacio vectorial continuo. Estos vectores capturan relaciones semánticas y sintácticas entre palabras, lo que significa que las palabras con significados o patrones de uso similares tendrán vectores que están cerca unos de otros.

Los *word embeddings* se aprenden típicamente de grandes cantidades de datos de texto, Hay dos enfoques principales:

1. **Modelos basados en predicción**: Estos modelos entrenan redes neuronales para predecir una palabra en función de su contexto circundante (o viceversa). Los pesos aprendidos de las capas ocultas de estas redes se convierten en los *word embeddings*.

2. **Modelos basados en recuento:** Estos modelos utilizan técnicas de factorización de matrices en matrices de co-ocurrencia para derivar *word embeddings*.

Los *Word embeddings* capturan el significado de las palabras, lo que permite tareas como encontrar sinónimos, antónimos o analogías.

Otra característica es que comprimen la representación dispersa y de alta dimensión de las palabras en vectores densos y de menor dimensión, lo que hace que los cálculos sean más eficientes.