



**ÍNDICE**

**Introducción**

**Lectura de datos**

**Aprendizaje de vocabulario**

**Ventana deslizante**

**One hot encoding (palabras del contexto y central)**

**Red neuronal (obtención de embeddings)**

**Obtención de diccionarios X\_train, X\_test, y\_train e y\_test**

**Obtención matriz de traducción R (por gradiente)**

**KNN**

**Resultados**

**INTRODUCCIÓN**

Este trabajo tiene como objetivo estudiar los conceptos tratados en la asignatura y aplicarlos para formar un traductor.

Nosotros hemos decidido traducir palabras de inglés a español ya que son dos de los idiomas más usados del mundo. Para ello lo primero que hemos hecho ha sido una búsqueda de ejemplos de frases en los dos lenguajes para que el traductor pueda aprender de ello. Nuestra elección han sido los libros de Harry Potter tanto en inglés como en español, de esta manera tenemos un léxico más o menos parecido ya que se tratan los mismos conceptos pero en dos idiomas distintos, además de que tenemos gran cantidad de oraciones con las que trabajar. De hecho tenemos tanto material que en un inicio pensamos en proporcionarle al algoritmo toda la saga de libros compuesta por siete ejemplares, pero al final hemos decidido usar un libro en cada idioma por el coste computacional que supondría estudiar los siete libros.

**LECTURA DE DATOS**

Una vez ya tenemos los recursos con los que empezar nuestro proyecto necesitamos ¨limpiarlos¨ o depurarlos de tal manera que se ajusten a las características del problema.

Para empezar quitamos todos los números, cambiamos los signos de puntuación que denotan final de frase por un punto, si salen varios seguidos solo ponemos un punto, y finalmente quitamos todo lo que no sean letras o puntos.

**APRENDIZAJE DE VOCABULARIO**

Para el aprendizaje del vocabulario, una vez hemos leído el texto y hemos preprocesado su contenido lo que tenemos que hacer es tokenizar el contenido del texto para obtener un listado con todas las palabras del mismo. Debemos tener en cuenta que en un futuro nos interesará trabajar con frases y para ello debemos almacenar algún símbolo del texto que delimite los inicios y finales de frase que en nuestro caso es el punto.

Para llevar a cabo esto el primer paso es utilizar algún tokenizador ya implementado que encaje con lo que buscamos, para ello vamos emplear el “word tokenize” de la librería de “nltk” que tokeniza por palabras. Una vez hecho esto como ya hemos dicho nos interesa guardar palabras que van a conformar las diferentes frases del texto y los puntos que van delimitar esas frases, por ello vamos a quedarnos con todos aquellos tokens que sean palabras que no contengan símbolos extraños ni números y también nos quedaremos con los tokens que sean un punto.

Este proceso lo vamos a aplicar tanto para nuestro corpus en español como para nuestro corpus en inglés, y ya tendremos nuestros vocabularios completos y depurados.

**CARACTERES ESPECIALES**

Con los libros ya preparados obtuvimos las palabras sin repeticiones que aparecen, por lo que ya tenemos una especie de vocabulario usado en ambos idiomas. Pero puede que no tengan las mismas palabras, de hecho es lo más probable y en nuestro caso pasa porque de un idioma a otro las expresiones y gramática cambian. Por lo que tenemos que hacer de alguna manera que las palabras en inglés tengan su traducción exacta y literal al español. Para ello lo que hicimos fue coger el vocabulario de inglés y con un traductor ya implementado como es el de google fuimos obteniendo su salida ideal. Esto lo usaremos más adelante tanto para comprobar si la traducción es correcta como para generar la matriz de cambio de lenguaje.

Como este procedimiento es un poco pesado nos guardamos estos datos en un fichero que luego podamos cargar más rápidamente que recalcular las palabras:



**VENTANA DESLIZANTE**

En este paso tendremos que elegir la forma de la que vamos a obtener nuestro embeddings y la forma de la que lo haremos será empleando un método clásico conocido como CBOW, el cual consiste en predecir una palabra en base a las palabras de su contexto, para ello más adelante implementaremos una red neuronal la cual nos devuelva una matriz de pesos que serán nuestros embeddings.

Para implementar el método CBOW lo que vamos a hacer es utilizar una ventana deslizante que va a recorrer las frases de nuestro corpus. Tener en cuenta que vamos a trabajar con frases es importante ya que es la unidad sobre la que vamos a operar. Para ello lo que haremos será dividir nuestro texto en frases mediante un “split” por puntos, de esta forma cada punto delimita el fin de una frase y el inicio de la siguiente.

Trabajar con una ventana deslizante implica que no siempre vamos a tener palabras de contexto para todas las palabras, ya que por ejemplo la primera palabra de la frase, no tiene palabras que vayan antes que ella y por tanto no podrá ser una palabra central , lo mismo ocurre con la última palabra de cada frase. Por ello tendremos que quedarnos solo con las palabras que si tengan el contexto completo, esas palabras formaran nuestro listado de palabras centrales, y las palabras del contexto formarán el listado de palabras contexto

**ONE HOT ENCODING**

Ahora que ya tenemos el listado de palabras centrales y palabras contexto debemos codificarlas para que más adelante la red neuronal pueda trabajar con ellas, para ello lo que vamos a utilizar es one hot encoding.

Inicialmente entrenaremos el one hot encoder de la librería de “catergory\_encoder” con las palabras del vocabulario, acto seguido haremos la transformación one hot de cada una de las palabras centrales, y para cada una de esas palabras centrales tendremos que hallar el correspondiente vector one hot de las palabras del contexto.

Como el contexto estará formado por varias palabras pero solo necesitamos un vector one hot lo que haremos será sumar los vectores one hot de las palabras del contexto y dividir ese vector entre el número de palabras del contexto, obteniendo así el vector de las palabras del contexto.

Con esto ya tendremos un vector one hot de las palabras del contexto y su correspondiente vector one hot de la palabra central.

**RED NEURONAL PARA EMBEDDINGS**

Una vez que tenemos el One Hot tanto para los conjuntos de palabras del contexto como las palabras centrales para cada contexto. Hay que entrenar una red neuronal para cada idioma para poder obtener los embeddings de las palabras para cada idioma.

La arquitectura de la red neuronal es algo especial para poder crear los embeddings, no se trata de red muy profunda. Puesto que solo tiene 3 capas en total:

* Capa de entrada:
  + Tiene el mismo número de neuronas que palabras en el vocabulario del idioma de la red neuronal.
* Capa oculta:
  + El número de neuronas determina el tamaño que tendrán los embeddings de los idiomas. Al modificar el número de neuronas de la capa oculta se obtendrán embeddings de distintos tamaños y así poder estudiar cómo afectan los tamaños de los embeddings de los idiomas a proceso de traducción.
  + Se aplica a la salida (la regresión lineal que se realiza ‘dentro’ de la neurona con los pesos) la función de activación RELU
* Capa de salida:
  + Vuelve a tener el mismo número de neuronas que el vocabulario para poder saber la predicción del vector de la palabra central para el contexto pasado como entrada.
  + A la capa de salida le aplicamos la función de softmax para que la probabilidad de cada neurona de la salida, que representan una palabra del vocabulario, su suma sea 1.

Para la creación de la arquitectura de la red neuronal empleamos las librerías de Tensorflow y Keras respectivamente, puesto que dejan configurar una red neuronal desde cero de forma sencilla. Además contando con que tendríamos un número elevado de palabras en cada vocabulario y también muchos conjuntos de palabras de contexto y centrales decidimos entrenar las redes neuronales empleando aceleración por tarjeta gráfica empleando las herramientas de Nvidia tales como CUDA y cuDNN.

El error que se emplea para medir la eficacia del entrenamiento de la red neuronal es de cross-entropy, además para acelerar el entrenamiento de la red neuronal se puede entrenar por lotes de ejemplos a si no se realiza el feed-forward y el backpropagation por cada ejemplo si no por cada lote, en Tenzoflow es muy sencillo de realizar y no es necesario modificar la arquitectura de la red neuronal para cada tamaño del lote.

Una vez entrenada la red neuronal varias veces con todos los ejemplos, otro parámetro que proporciona Tensorflow para mejorar la precisión de la propia red neuronal. Podemos obtener los embeddings de las palabras de varias formas empleando los pesos entrenados de la red neuronal:

* Empleando los pesos de la capa entrada a la oculta
* Empleando los pesos de la capa oculta a la de salida
* Combinando ambas

Nosotros empleamos los pesos de la capa de oculata (W2), que tiene un tamaño de *palabras vocabulario x neurona capa oculta.*

**FILTRADO DE EMBEDDINGS**

Los vocabularios del libro en inglés y en español no son los mismos, por lo que tenemos que ver qué palabras salen en los dos libros para generar X e Y y poder aplicar el descenso por gradiente con el que sacar la matriz de cambio de lenguaje. Para ello usamos el documento que guardamos al principio del proyecto con las traducciones literales. Lo que hacemos es ver qué palabras están tanto en la traducción literal del vocabulario en inglés y en el vocabulario original en español. Esas palabras las guardamos en los dos idiomas y reservamos un porcentaje, el 20% en nuestro caso para el test.

Los embeddings correspondientes a las palabras que nos hemos guardado en español para train será la matriz X y lo mismo pero en inglés será la matriz Y. De esta forma ya estamos en condiciones para formar la matriz R de cambio de base.

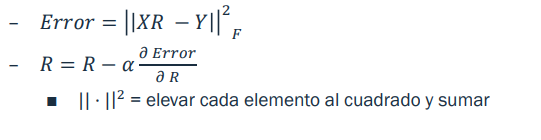
**OBTENCIÓN MATRIZ DE TRADUCCIÓN**

Para poder obtener la traducción de nuestras palabras en inglés vamos a tener que usar una matriz de traducción.

Esta matriz de traducción lo que va a hacer es dado el embedding de la palabra va a obtener su embedding correspondiente en el idioma al que queremos traducir, en este caso español.

Para poder hacer esto, esta matriz debe haberse entrenado previamente con nuestros embeddings en inglés y español del conjunto de train, es por ello que mediante un proceso de regresión haremos que nuestra matriz R aprenda el paso de palabras de inglés a castellano mediante sus embeddings.

La regresión se llevará a cabo de la siguiente manera:



1. Primero deberemos definir una función de error, este error será la suma de las diferencias entre los embeddings que hemos hallado y los que deberíamos haber hallado. Esta función hará que al tratar de minimizar en el proceso de regresión cada vez nos aproximamos más a los embeddings que realmente queremos obtener.
2. Debemos definir un valor alfa, que será el tamaño del salto que va a sufrir la matriz de una iteración a la siguiente, un valor de alfa muy alto puede dificultar la convergencia de la matriz R y hacer que empeore y un valor demasiado pequeño puede ralentizar excesivamente el proceso de convergencia.
3. Por último debemos hallar la derivada de la función de error respecto de nuestra matriz R.

Una vez definidos estos parámetros solo tenemos que buscar un número de iteraciones suficiente como para que la matriz R llegue a un buen resultado.

Para hacer posible el producto de matrices en el proceso de gradiente, la matriz R debe ser de tamaño NxN siendo N el tamaño que hayamos definido para los embeddings, además esta matriz en un principio no tendrá valores o seran todos 0 es por ello que la debemos inicializar aleatoriamente, esto no afectará demasiado al resultado ya que en el proceso de gradiente la tendencia seguirá siendo a disminuir el error.

**KNN**

Empleando la matriz de traducción R, empleamos los embeddings de las palabras en español que queramos traducir. Multiplicamos esos embeddings a traducir por la matriz R para pasar al espacio vectorial de los embeddings en español. Teniendo este nuevo punto en el espacio vectorial tenemos que ver cuál es su palabra más parecida.

Para ello empleamos el modelo de inteligencia artificial Knn para poder obtener la palabra más parecida al punto calculado. El Knn se entrena con los embeddings de español siendo la entrada para el clasificador y con la lista de las palabras en el vocabulario de español, que son la salida y lo que se quiere predecir posteriormente.

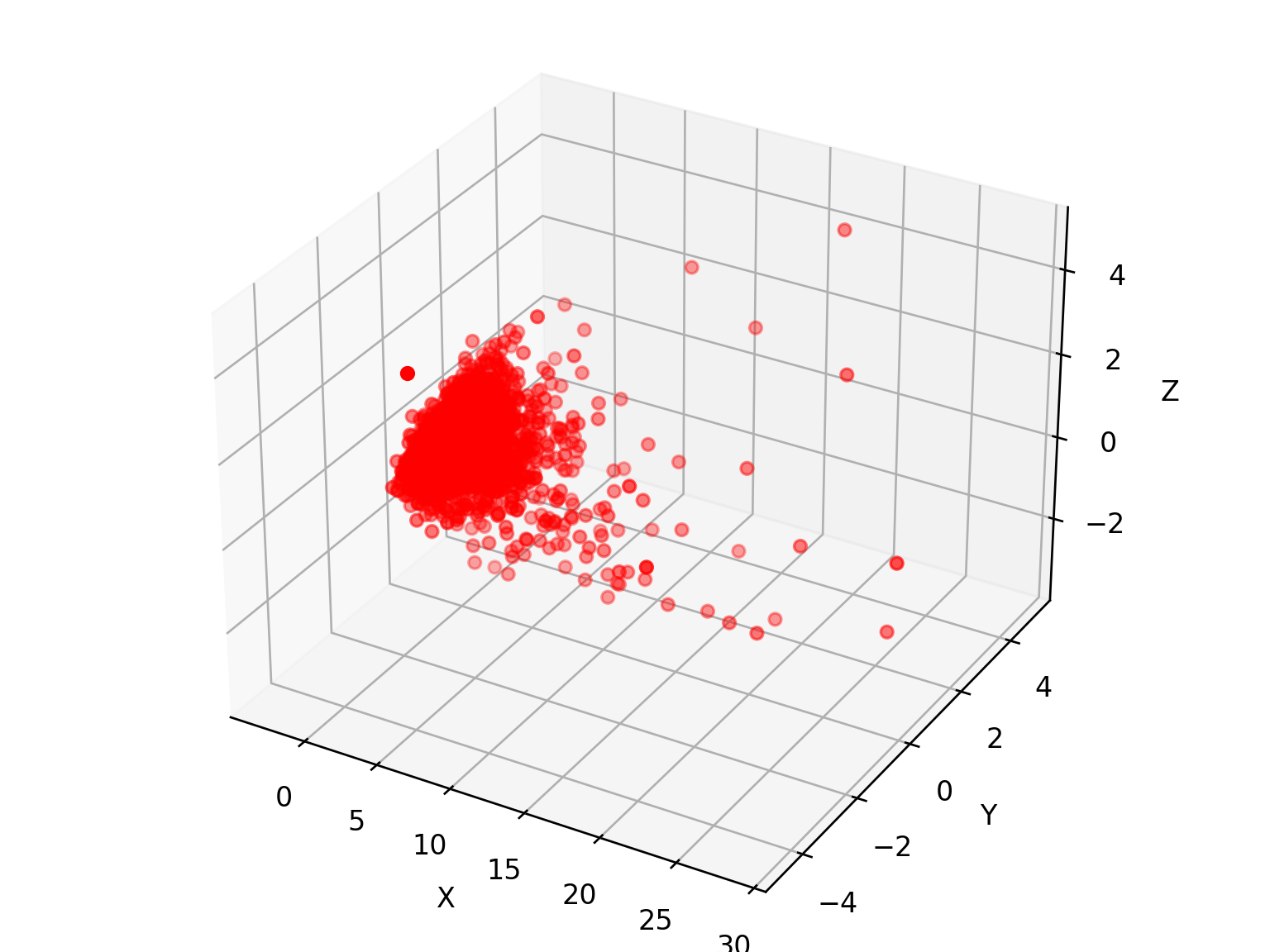
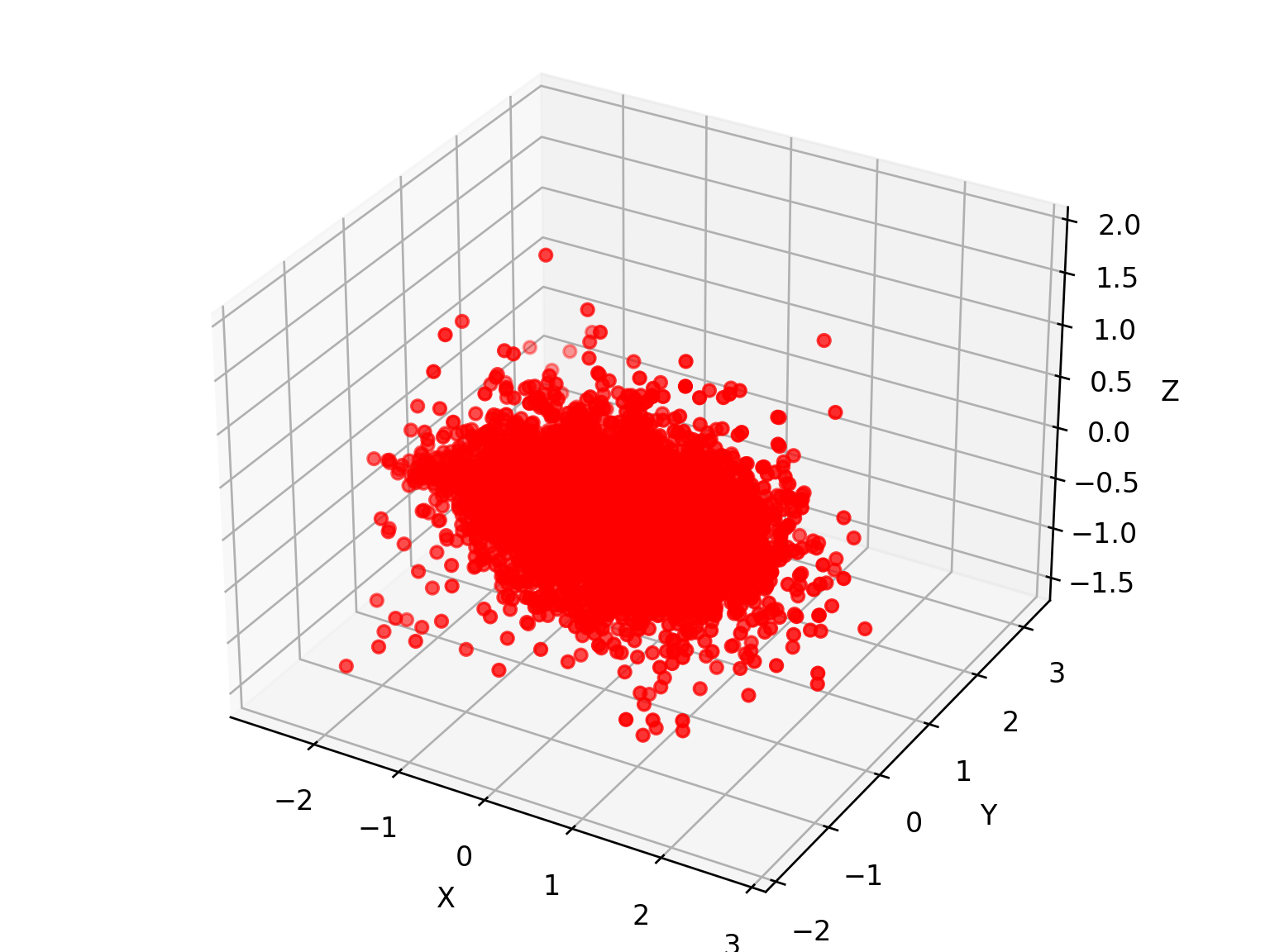
Como queremos obtener la palabra que más se parezca al punto del nuevo espacio vectorial donde tenemos que obtener el embedding más cercano, para ello empleamos el atributo número de vecinos a uno.

**RESULTADOS**

Una vez terminado el proceso de aprendizaje podemos ver cómo de bien traducimos:



Como podemos ver los resultados no son especialmente buenos y creemos que puede ser porque los embeddings salen muy parecidos o incluso se superponen, lo cual explicaría que muchas de las palabras sugeridas sean cama. Para ello hemos impreso en 3D los embeddings y este es el resultado:

➜

**CONCLUSIÓN**

Como conclusión no podemos decir si los embeddings son buenos o no ya que al realizar el traductor no ha salido como desearíamos, pero tendríamos que hacer una evaluación intrínseca como un test de analogías por ejemplo para poder decir definitivamente que son malos embeddings. Hemos probado a visualizar los embeddings en el plano pero salen muy apelotonados y se dificulta la interpretabilidad, además al tener que representarlo en máximo 3 dimensiones perdemos mucha información.