

# Estudio e implementación de los enfoques de embeddings multilingües de palabras

Danilo Blas Salas  
Universidad Nacional de Ingeniería  
Lima, Perú  
iblass@uni.pe

Bryan Asmat Franco  
Universidad Nacional de Ingeniería  
Lima, Perú  
basmatf@uni.pe

**Resumen**—En lingüística, las incrustaciones de palabras se discutieron en el área de investigación de la semántica distributiva. Su objetivo es cuantificar y categorizar las similitudes semánticas entre elementos lingüísticos en función de sus propiedades de distribución en grandes muestras de datos lingüísticos. La idea subyacente de que una palabra se caracteriza por la compañía que mantiene fue popularizada por John Rupert Firth. Por otro lado, Una embedding de palabras es una representación aprendida de un texto, donde las palabras que tienen el mismo significado tienen una representación similar. Este enfoque de la representación de palabras y documentos puede ser considerado como uno de los principales avances del Deep Learning en los problemas de procesamiento del lenguaje natural. En el presente informe abarcaremos diferentes conceptos previos sobre los modelos de semántica distribuida realizando que serán acompañados por su implementación realizada en python abarcando temas como: Embeddings de palabras multilingües, embeddings de subpalabras y embeddings de oraciones.

## I. INTRODUCCIÓN

Los últimos años han sido un momento emocionante para estar en el campo del NLP (*natural language processing*). La evolución de los vectores de palabras basados en frecuencias escasas a los modelos de representación de palabras semánticas densas, como Word2vec y GloVe [7], sentó las bases para aprender el significado de las palabras. Durante muchos años, sirvieron como inicializaciones confiables de la capa de incrustación para entrenar modelos en ausencia de grandes cantidades de datos específicos de la tarea. Dado que los modelos de incrustación de palabras previamente entrenados en Wikipedia, estaban limitados por el tamaño del vocabulario o la frecuencia de apariciones de palabras.

## II. REPRESENTACIÓN DE PALABRAS

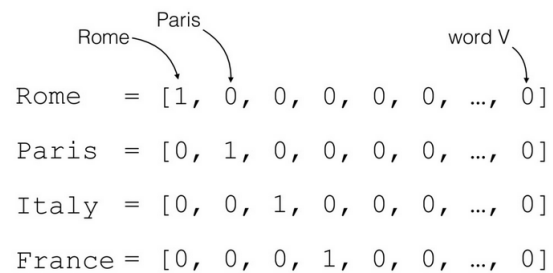
### II-A. Codificación One-Hot

Una de las formas básicas en que se puede representar palabras numéricamente es mediante el método de codificación *One-Hot* (también llamado vectorización de conteo).

La estrategia que implementa es crear una columna para cada valor distinto que exista en la característica que estamos codificando y, para cada

registro, marcar con un 1 la columna a la que pertenezca dicho registro y dejar las demás con 0.

El resultado son Vectores enormes y escasos que no capturan absolutamente ninguna información relacional. Podría ser útil si no tiene otra opción. Pero existen otras opciones si se requiere información semántica.



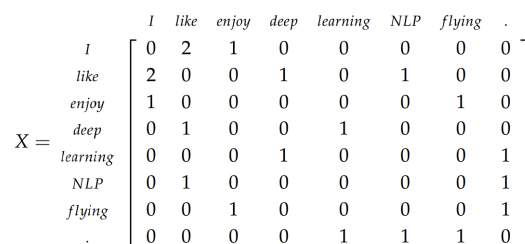
Rome	=	[1, 0, 0, 0, 0, 0, ..., 0]
Paris	=	[0, 1, 0, 0, 0, 0, ..., 0]
Italy	=	[0, 0, 1, 0, 0, 0, ..., 0]
France	=	[0, 0, 0, 1, 0, 0, ..., 0]

Figura 1. Representación vectorial de palabras del modelo *One-Hot* [1]

### II-B. Matriz de coocurrencia

Una matriz de co-ocurrencia es una matriz que es tan larga y tan ancha como el tamaño del vocabulario. Si las palabras aparecen juntas, se marcan con una entrada positiva. De lo contrario, tienen un 0.

El problema con la matriz de coocurrencia es su tamaño (su dimensión es el tamaño del vocabulario al cuadrado), ya que es una gran cantidad de datos para almacenar en la memoria para vocabularios muy grandes.



	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Figura 2. Ejemplo de la matriz de coocurrencia con un vocabulario simple [2]

### III. MODELOS DE LENGUAJE

#### III-A. Modelo probabilístico neuronal

Un modelo probabilístico neuronal aprende un *embedding* al lograr alguna tarea como modelado o clasificación, y es en lo que se basan el resto de los *embedding*.

Por lo general, limpia su texto y crea vectores codificados en un solo uso. Luego, define el tamaño de su representación. A partir de ahí, se inicializa el *embedding* a valores aleatorios. Es el punto de entrada a la red y la propagación hacia atrás se utiliza para modificar el *embedding* en función de cualquier objetivo que se tenga.

Por lo general, esto requiere una gran cantidad de datos y puede ser muy lento. La compensación, es que aprende una incorporación que es buena para los datos de texto en los que se entrenó la red, así como para la tarea de NLP que se aprendió conjuntamente durante el entrenamiento [3].

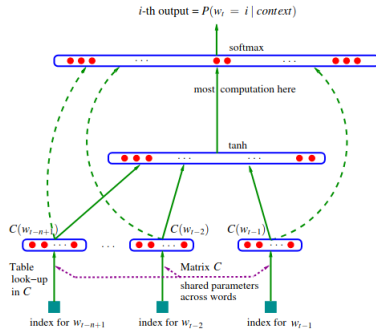


Figura 3. Arquitectura neuronal:  $f(i, w_{t-n+1}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+n-1}) = g(i, C(w_{t-n+1}), \dots, C(w_{t+1}))$  donde  $g$  es la red neuronal y  $C(i)$  es el  $i$ -ésimo vector característico de la palabra. [3]

#### III-B. Modelo Modelo word2vec

Se propuso la herramienta word2vec para abordar el problema anterior. Asigna cada palabra a un vector de longitud fija, y estos vectores pueden expresar mejor la relación de similitud y analogía entre diferentes palabras. La herramienta word2vec contiene dos modelos, a saber, skip-gram [Mikolov et al., 2013b] y *Continuous bag-of-words* (CBOW) [Mikolov et al., 2013a]. Para las representaciones semánticamente significativas, su entrenamiento se basa en probabilidades condicionales que pueden verse como la predicción de algunas palabras usando algunas de las palabras que las rodean en corpus. Dado que la supervisión proviene de los datos sin etiquetas, tanto el salto de gramo como el paquete continuo de palabras son modelos auto-supervisados.

#### III-C. Continuous Bag-of-Words (CBOW)

La tarea falsa en CBOW es algo similar a Skip-gram, en el sentido de que todavía tomamos un par de palabras y le enseñamos al modelo que

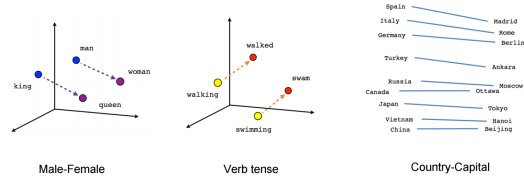


Figura 4. [5]

coexisten, pero en lugar de agregar los errores agregamos las palabras de entrada para la misma palabra objetivo.

La dimensión de nuestra capa oculta y la capa de salida seguirá siendo la misma. Solo cambiará la dimensión de nuestra capa de entrada y el cálculo de las activaciones de la capa oculta, si tenemos 4 palabras de contexto para una sola palabra de destino, tendremos 4 vectores de entrada  $1 \times V$ . Cada uno se multiplicará con la capa oculta  $V \times E$  que devuelve los vectores  $1 \times E$ . Los 4 vectores  $1 \times E$  se promediarán por elementos para obtener la activación final que luego se alimentará a la capa softmax.

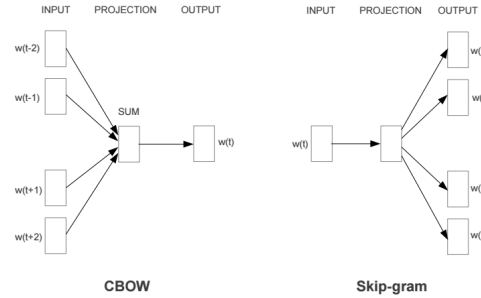


Figura 5. [4]

#### III-D. GloVe: Global Vectors for Word Representation

Es un híbrido de modelos basados en recuento y basados en ventanas. La ventaja de GloVe es que, a diferencia de Word2vec, GloVe no se basa solo en estadísticas locales (información de contexto local de palabras, modelos basados en ventanas), sino que incorpora estadísticas globales (co-ocurrencia de palabras, modelos basados en recuento) para obtener vectores de palabras [7].

##### Función Objetivo:

$$J(\theta) = 1/2(\sum_{i,j=1}^W f(P_{i,j})(u_i^T v_j - \log P_{i,j})^2$$

Por cada par de palabras  $i$  y  $j$  que puedan coexistir, tratamos de minimizar la diferencia entre el producto interno de su incrustación de palabras y el recuento logarítmico de  $i$  y  $j$ . El término  $f(P_{ij})$  nos permite ponderar algunas co-ocurrencias muy

frecuentes y limitar la importancia de palabras muy frecuentes.

Los vectores  $u$  y  $v$  aquí son intercambiables. Usamos dos conjuntos diferentes de vectores aquí porque proporciona más estabilidad durante la optimización. Si usáramos un solo conjunto de vectores, realizaremos un producto interno consigo mismo y tendremos una función objetiva de menor comportamiento. En nuestra evaluación final, simplemente combinaremos los dos realizando una adición inteligente de elementos.

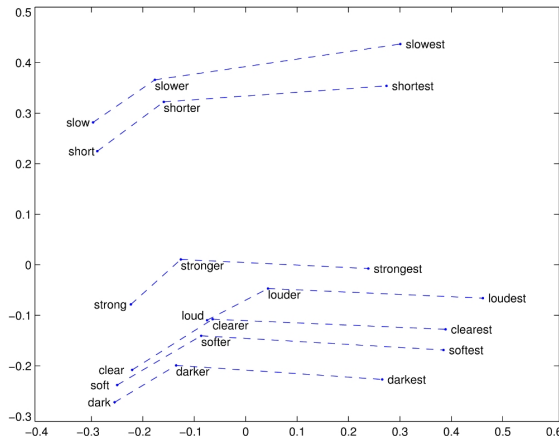


Figura 6. [4]

#### IV. EMBBETING DE PALABRAS

Cuando hablamos de procesamiento del lenguaje natural (NLP), estamos discutiendo la capacidad de un modelo de aprendizaje automático para conocer el significado del texto por sí solo, y realizar ciertas funciones similares a las humanas, como predecir la siguiente palabra u oración, escribir un ensayo basado en la información dada, o para conocer el sentimiento detrás de la palabra o un párrafo.

Algunas técnicas que conozcamos como *Bag of Words* son buenas para datos a pequeña escala, pero carecen en la extracción de significado de una oración o párrafo; además su destreza es limitada ya que no le importa el orden de las palabras.

El embbeting de palabras (*word embedding*) resuelve la mayoría de los problemas

##### IV-A. ¿Por qué embbeting de palabras ?

Lo que hace que el *embedding* de palabras sea diferente y poderosa de otras técnicas es que funciona con las limitaciones de otras, como *Bag of words*. Algunos puntos que hacen que el embedding de palabras sea mejor que otras son:

- Una mejor comprensión de las palabras y oraciones que otras técnicas de NLP, también conocida como análisis lingüístico.
- Reduce las dimensiones del conjunto de datos mejor que otras técnicas de NLP y, por lo tanto, funciona mejor.

- Requiere menos tiempo de ejecución o, en otras palabras, es más rápido en el entrenamiento que otros, ya que no requiere una enorme cantidad de pesos para entrenar como otros.
- No sigue el enfoque de la matriz dispersa, por lo que la hace mejor por razones computacionales.

##### IV-B. *embedding multilingüe*

Multilingual Word Embeddings (MWEs) representan palabras de varios idiomas en un único espacio vectorial de distribución. Los métodos MWE no supervisados (UMWE) adquieren integraciones multilingües sin supervisión multilingüe, lo que es una ventaja significativa sobre los enfoques supervisados tradicionales y abre muchas posibilidades nuevas para los idiomas de bajos recursos. Sin embargo, la técnica anterior para el aprendizaje de UMWE se basa simplemente en una serie de incrustaciones de palabras bilingües no supervisadas (UBWE) formadas de forma independiente para obtener incrustaciones multilingües.

##### IV-C. *Embedding de subpalabras*

Las palabras en inglés suelen tener estructuras internas y métodos de formación. Por ejemplo, podemos deducir la relación entre "dog", "dogs", "dogcatcher" por su ortografía. Todas estas palabras tienen la misma raíz, "dog", pero usan diferentes sufijos para cambiar el significado de la palabra. Además, esta asociación puede extenderse a otras palabras. Por ejemplo, la relación entre "dogz", "dogs.es" como la relación entre "çatz", "çats". La relación entre "boyz", "boyfriend.es" como la relación entre "girlz", "girlfriend". Esta característica no es exclusiva del inglés. En francés y español, muchos verbos pueden tener más de 40 formas diferentes según el contexto. En finlandés, un sustantivo puede tener más de 15 formas. De hecho, la morfología, que es una rama importante de la lingüística,

*IV-C1. fastText:* fastText propone el método de incrustación de subpalabras, intentando así introducir información morfológica en el modelo skip-gram en word2vec. Cada palabra central se representa como una colección de subpalabras.

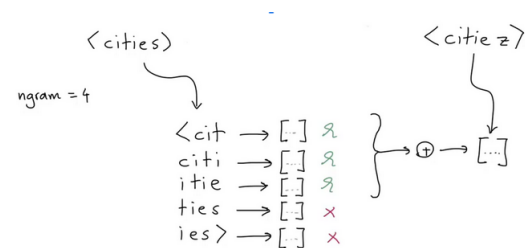


Figura 7. Ejemplo gráfico de la metodología de *fastText* para relacionar palabras

#### IV-D. *Embeddings de oraciones*

Las técnicas de inserción de oraciones representan oraciones completas y su información semántica como vectores. Esto ayuda a la máquina a comprender el contexto, la intención y otros matices de todo el texto.

Al igual que la inserción de palabras, la inserción de frases también es un área de investigación muy popular con técnicas muy interesantes que rompen la barrera para ayudar a la máquina a comprender nuestro idioma.

- Doc2Vec
- SentenceBERT
- InfeSent
- Universal Sentence Encoder

### V. NATURAL LANGUAGE PROCESSING NLP

#### V-A. *Byte Pair Encoding (BPE)*

Dejando a un lado las incrustaciones a nivel de caracteres, los investigadores de la Universidad de Edimburgo lograron el primer avance real para abordar el problema de las palabras raras al aplicar unidades de subpalabras en la traducción automática neuronal utilizando la codificación de pares de bytes (BPE). Hoy en día, los esquemas de tokenización de subpalabras inspirados en BPE se han convertido en la norma en la mayoría de los modelos avanzados, incluida la muy popular familia de modelos de lenguaje contextual como BERT, GPT-2, RoBERTa, etc.

Muchos algoritmos de compresión reemplazan los patrones de bits que ocurren con frecuencia con representaciones más cortas. El enfoque BPE es reemplazar pares comunes de bytes por bytes individuales.

El algoritmo comprime los datos, encontrando los pares de bytes adyacentes que ocurren con mayor frecuencia en los datos y reemplazar todas las instancias con un byte que no estaba en los datos originales. El algoritmo repite este proceso hasta que no es posible una compresión adicional, ya sea porque no hay pares que ocurren con más frecuencia o porque no hay más bytes sin usar para representar pares. El algoritmo escribe la tabla de sustituciones de pares antes de los datos empaquetados.

Este algoritmo es fundamentalmente de múltiples pasadas y requiere que todos los datos se almacenen en la memoria. Este requisito causa dos problemas potenciales: el algoritmo no puede manejar flujos y algunos archivos pueden ser demasiado grandes; y, es posible que los archivos binarios grandes no contengan caracteres no utilizados para representar sustituciones de pares.

El almacenamiento en búfer de pequeños bloques de datos y la compresión de cada bloque por separado resuelve estos problemas. El algoritmo lee y almacena datos hasta que el búfer está lleno

o solo queda un número mínimo de caracteres no utilizados. Luego, el algoritmo comprime el búfer y genera el búfer comprimido junto con su tabla de pares. El uso de una tabla de pares diferente para cada bloque de datos también proporciona una adaptación local a datos variables y mejora la compresión general.

*V-A1. BPE para la tokenización de subpalabras* : Para realizar la tokenización de subpalabras, BPE se modifica ligeramente en su implementación de modo que los pares de subpalabras que ocurren con frecuencia se fusionan en lugar de ser reemplazados por otro byte para permitir la compresión.

#### Algoritmo

- Paso 0: Inicializar el vocabulario
- Paso 1: Representar cada palabra en el corpus como una combinación de los caracteres junto con el token de fin de palabra (</w>).
- Paso 2: Contar iterativamente los pares de caracteres en todas las fichas del vocabulario.
- Paso 3: Combinar cada aparición del par más frecuente y agregar el nuevo carácter n-grama al vocabulario.
- Paso 4: Se repite el paso 3 hasta que se complete el número deseado de operaciones de combinación o se alcance el tamaño de vocabulario deseado (que es un hiperparámetro)

### VI. DISCUSIÓN E INTERPRETACIÓN DE RESULTADOS

- El preprocesado de texto es crucial para el desarrollo de las técnicas de NLP dado que cuanto más conjunto de datos o corpus, más entrenamiento se requiere, sin embargo, la limitación radica en el crecimiento significativo en el espacio de la memoria.
- El uso de las diversas librerías con modelos preentrenados nos da facilidad para el uso de diferentes aplicativos usando embedding word sobre todo en palabras multilingües.
- Una de las principales limitaciones de las incrustaciones de palabras (modelos de espacio vectorial de palabras en general) es que las palabras con múltiples significados se combinan en una sola representación (un solo vector en el espacio semántico). En otras palabras, la polisemia y la homonimia no se manejan adecuadamente.

#### REFERENCIAS

- [1] [https://cdn-images-1.medium.com/max/1600/1\\*YEJf9BQZh0ma1ECs6x7yQ.png](https://cdn-images-1.medium.com/max/1600/1*YEJf9BQZh0ma1ECs6x7yQ.png)
- [2] <https://i.stack.imgur.com/oJEie.png>
- [3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent y Christian Jauvin . A Neural Probabilistic Language Model (2003). Disponible en <https://www.jmlr.org/papers/volume3/tmp/bengio03a.pdf>.
- [4] <https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/06/06062705/Word-Vectors.png>

- [5] <https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/06/06062705/Word-Vectors.png>
- [6] <https://www.researchgate.net/profile/Kiril-Simov-2/publication/312045706/figure/fig7/AS:446766601510921@1483528866667/The-c-bow-and-skip-gram-architectures-figure-taken-from-9.png>
- [7] Jeffrey Pennington, Richard Socher, Christopher D. Manning. GloVe: Global Vectors for Word Representation. Disponible en <https://nlp.stanford.edu/pubs/glove.pdf>.
- [8] Sebastian Ruder, Ivan Vulić , Anders Søgaard. A Survey of Cross-lingual Word Embedding Models. Disponible en <https://core.ac.uk/download/pdf/162916493.pdf>.