Universidad de Valladolid

FACULTAD DE CIENCIAS

Aplicación de métodos NLU en la recomendación de CVs para la selección de personal

GRADO EN ESTADÍSTICA E INVESTIGACIÓN OPERATIVA

Autora:

Sofía Mara Rivas Cuevas

Tutora:

Paula Gordaliza Pastor

Trabajo de fin de grado

Universidad de Valladolid

Julio 2022

Sic Parvis Magna
Sir Francis Drake

Agradecimientos

A mi familia, sobre todo a mis padres, Graciela y Ariel, y a mi hermano, Agustín, por su apoyo incondicional y su aguante para soportarme tanto en mis mejores momentos como en mis horas más bajas. Os quiero mucho y os lo agradezco infinitamente.

A mi pareja, Roberto, por estar pendiente, cuidarme y tener siempre amor y cariño para mí.

A mi tutora, Paula, por su tiempo y por guiarme durante este camino. Gracias por estar siempre ahí y por implicarte tanto.

A todos mis compañeros de carrera, gracias por estos años y por vuestro apoyo y ayuda.

Y por último gracias a mí misma por no rendirme nunca y por creer en mí.

Resumen

Hace años, a la hora de buscar empleo, los trabajadores abrían el periódico y dirigían su mirada a la columna dedicada a la búsqueda de personal, seleccionando la ocupación que mejor se adaptara a su perfil. Ahora, con el auge de la Inteligencia Artificial, no es de extrañar que este proceso también se haya modernizado, desarrollándose sistemas de recomendación para la búsqueda de personal.

En este trabajo de fin de grado se desarrollará un sistema de recomendación de CVs con el fin de dotar de una mayor agilidad al proceso de selección de personal. Se dispondrán de datos de CV y de ofertas laborales, y como resultado se obtendrán los mejores candidatos para cada puesto. A mayores a los candidatos no seleccionados se les propondrá mejorar sus aptitudes técnicas mediante la recomendación de cursos formativos. Se utilizarán técnicas de Procesamiento del Lenguaje Natural (NLP) y de Entendimientos del Lenguaje Natural (NLU) para lograr el objetivo marcado, así como la utilización de una métrica de la adecuación de los candidatos a la oferta, fundamentada en el uso de la similitud del coseno.

Abstract

Years ago, when looking for a job, workers opened the newspaper and directed their gaze to the column dedicated to the search for personnel, selecting the occupation that best suited their profile. Now, with the rise of Artificial Intelligence, it is not surprising that this process has also been modernized, developing recommendation systems for the search for personnel.

In this end-of-degree project, a CV recommendation system will be developed in order to provide greater agility to the personnel selection process. We will have CV data and job offer data and as a result we will obtain the best candidates for each position. In addition, unselected candidates will be offered to improve their technical skills through the recommendation of training courses. Natural Language Processing (NLP) and Natural Language Understanding techniques will be used to achieve the target marking, as well as the use of a metric of the adequacy of the candidates to the fundamental offer in the use of cosine similarity.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

Los datos son empleados diariamente por las empresas para obtener información sobre sus clientes. Su finalidad es predecir modas futuras, encontrar nuevas estrategias de marketing para acercar su producto a los consumidores potenciales o recomendar artículos adaptados a nuestro perfil. En muchas ocasiones se pretende procesar toda esa información para darle al usuario una atención más personalizada. De esta necesidad surgen los recomendadores, sistemas que pretenden administrar y procesar nuestros datos para poder recomendar los elementos que mejor se adapten a nuestro perfil. Los sistemas de recomendación son, en la actualidad, un elemento que empleamos a diario, bien para decidir que producto adquirir y directamente la aplicación nos recomienda el adecuado basado en nuestras compras pasadas o en perfiles similares al nuestro, o bien para seleccionar una película del catálogo de Netflix.

En el caso concreto de una empresa, que recibe una gran cantidad de currículums y que por consiguiente procesa una gran cantidad de información sobre cada candidato, la necesidad de procesar los datos obtenidos, clasificarlos y posteriormente seleccionar al candidato idóneo para cada puesto en un tiempo razonable, supone todo un reto en la mayoría de los casos, muchas veces inasumible por el ser humano. Puesto que muchas de las ocasiones en las que se selecciona personal se debe realizar todo este proceso en un tiempo reducido, se ha desarrollado esta herramienta que pretende agilizar el proceso de selección proporcionando un ranking de los currículums que más se adecuen a una determinada oferta laboral. Esta herramienta ha sido desarrollada en colaboración con la empresa Air Institute, donde implementaremos un sistema de recomendación de currículums (CVs) para las distintas ofertas laborales que una empresa tecnológica puede presentar. Como en todo proceso de selección de personal, se dispondrá de candidatos que si bien poseen ciertas habilidades requeridas para el puesto, no son los más idóneos para desempeñar esa labor, o bien por la falta de manejo de algún lenguaje de programación o por falta de conocimiento sobre cierto tema, para estos casos se ha implementado un segundo recomendador que

propondrá algunos cursos formativos que pueden ser de interés a realizar por el candidato para poder optar a ofertas similares en un futuro. Cabe aclarar que este sistema de recomendación no toma la decisión de que candidato debe ser seleccionado finalmente para el puesto, sino que pretende agilizar el proceso de selección utilizando técnicas de Inteligencia Artificial.

Para finalizar la introducción, se dará una breve descripción de los distintos capítulos que conforman este proyecto; En este documento se abordará primero la parte teórica del problema, en el cual describiremos diversos algoritmos de clustering que se han formulado para la solución del problema, así como las diferentes métricas que se pueden aplicar en cada uno. También comentaremos los distintos tipos de sistemas de recomendación y de sus aplicaciones en la vida cotidiana. Para finalizar la parte del marco teórico comentaremos el tema del Procesamiento del Lenguaje Natural o Programación Neurolingüística así como los pasos a seguir para poder utilizar esta técnica que como veremos su inicio se remonta a los años 50. A continuación se proporcionará una visión general del código que se ha empleado para implementar el recomendador así como una breve descripción del funcionamiento de cada clase. Por último, se expondrán los resultados que nos proporcione el algoritmo implementado con datos que nos permitan mantener la confidencialidad de los candidatos que optan a los distintos puestos. Finalmente el último capítulo de este documento corresponde al apartado de conclusiones, donde se comentaran las conclusiones extraídas después de realizar el proyecto, así como las posibles mejoras que se pueden realizar para optimizar el recomendador y posibles trabajos futuros que puedan sobrevenir.

Capítulo 2

Algoritmos de clasificación no supervisada de cadenas de texto

El aprendizaje automático o Machine Learning (de ahora en adelante se denotará como ML), es un sub-área de la Inteligencia Artificial (de ahora en adelante IA), encargada de desarrollar técnicas cuya finalidad es que los ordenadores o computadoras aprendan de manera automática. En lo que respecta a manejar una gran cantidad de datos, las distintas técnicas de aprendizaje automático resultarán de gran ayuda. En el artículo (Hinestroza Ramírez 2018) se comentan diversos campos en los que el aprendizaje automático ha supuesto un cambio en la forma de trabajar, siendo el más notable en el campo de la medicina. Donde se han desarrollado métodos con el objetivo de mejorar la eficiencia de los equipos generadores de diagnósticos médicos y evitar los posibles fallos que un ser humano pueda cometer a la hora de analizar e interpretar los datos.

Dentro del aprendizaje automático se disciernen tres sub-áreas; en primer lugar se describirá el **aprendizaje supervisado**, en este caso los algoritmos emplean datos que previamente han sido etiquetados. Este etiquetado indicará como tendrá que ser categorizada la nueva información. Paralelamente el **aprendizaje no supervisado** emplea paradigmas que no se apoyan en datos que han sido categorizados. Los modelos desarrollados con esta clase de aprendizaje se centrarán en identificar puntos en común entre las características, que sean capaces de clasificar la nueva información (Rebollo and Barrojo 2009). Por último, el **aprendizaje por refuerzo**, emplea modelos que aprenden a tomar la decisión más adecuada para lograr el objetivo. Estos algoritmos emplean procesos iterativos de prueba y error, al cual se le aplicará un sistema de refuerzos. Estos refuerzos pueden ser positivos, en el caso de la decisión tomada fuera la adecuada, o negativos, en su defecto.

Un problema recurrente en el Análisis Multivariante consiste en clasificar n individuos en k grupos o clusters distintos, de tal forma que los individuos dentro de un

mismo grupo sean homogéneos y entre los distintos grupos sean los más heterogéneos posibles. Este tipo de análisis, el análisis clúster, se encuentra entre los métodos que emplean aprendizaje no supervisado. Es el propio algoritmo es el que determina el número de grupos y los valores que identificarán a cada uno.
A continuación se proporcionarán ciertas definiciones que son relevantes para comprender mejor el Análisis Clúster:

(Todas las definiciones han sido extraídas del libro (Peña 2002))

Definición 1 (Distancia)**.** *Dados dos puntos x_i, x_j pertenecientes a $\mathbb{R}^p$, diremos que hemos establecido una distancia, o una métrica, entre ellos si hemos definido una función d con las propiedades siguientes:*

- $d : \mathbb{R}^p \to \mathbb{R}^+$*, es decir, dados dos puntos en el espacio dimensión p su distancia con esta función es un número no negativo,* $d(x_i, x_j) \geq 0$.
- $d(x_i, x_i) = 0 \ \ \forall_i$*, es decir, la distancia entre cada elemento y sí mismo es cero.*
- $d(x_i, x_j) = d(x_j, x_i)$*,es decir, la distancia es una función simétrica en sus argumentos.*
- $d(x_i, x_j) \leq d(x_i, x_p) + d(x_p, x_j)$*, la distancia debe verificar lo siguiente: si tenemos tres puntos, la suma de las longitudes de dos lados cualesquiera del triángulo formado por dichos puntos debe ser siempre mayor que el tercer lado. Esta propiedad se conoce como la propiedad triangular.*

Definición 2 (Similitud)**.** *La mayoría de autores definen la similitud como lo contrario de la distancia. Cuanto mayor sea el valor de la distancia, más lejos se encuentran los elementos y viceversa, en cambio, cuando hablamos de similitud cuanto mayor sea el valor, más parecidos son los elementos. Siendo el valor de la similitud 0 cuando los elementos son completamente distintos.*

Una vez conocidas estas definiciones, se procederá a describir los algoritmos básicos y más utilizados del Análisis Clúster.

Algoritmo K-medias *(K-means)*

Se dispone de una muestra con n individuos y p variables. El objetivo será clasificar dicha muestra en un número de grupos que se habrá elegido con anterioridad. Estos grupos se denominarán con la letra K, cuyos centros óptimos serán $C_1^*, ..., C_k^*$ en $\mathbb{R}^p$. Este algoritmo constará de cuatro etapas para su realización:

- En primer lugar se seleccionarán los centros de los grupos iniciales (K). Para determinar los centros de los grupos iniciales se pueden utilizar distintos procedimientos::

- Se asignarán de manera aleatoria los individuos a los grupos. Adoptándose los centros que se formen como los iniciales.
- Tomando los K puntos más alejados entre sí como los centros.
- Se podrán clasificar a los individuos empleando información que se disponga anteriormente es decir, a priori o de igual manera seleccionar los centros.

- A continuación se calcularán las distancias de cada uno de los individuos a los centros de los K grupos que se han seleccionado en el paso anterior. La asignación se realiza de manera secuencial. La métrica más empleada es la distancia euclídea. Al introducir un nuevo elemento se recalcularán las coordenadas de la nueva media del clúster.

- Se definirá un criterio de optimalidad para comprobar si reasignando un elemento de un clúster a otro el criterio mejora. En este caso el criterio de optimalidad o función de coste es:

$$C_1, C_2, ..., C_k = \arg\min \sum_{i=1}^{k} \sum_{x \in S_i} ||x - C_i||^2$$

  Minimiza la suma sobre cada clúster de la suma del cuadrado de la distancia entre el punto y su centroide. En caso de no mejorar el criterio de optimalidad, se dará por finalizado el algoritmo.

A cada punto se le asignará uno de los grupos mediante la reducción de la suma de cuadrados en el clúster, es decir, asignará los puntos al clúster más cercano, manteniendo de esta manera los clústers los más pequeños posibles. (Peña 2002) En las siguientes figuras se desarrollará un ejemplo del algoritmo K-Medias, con el fin de entender mejor su funcionamiento:

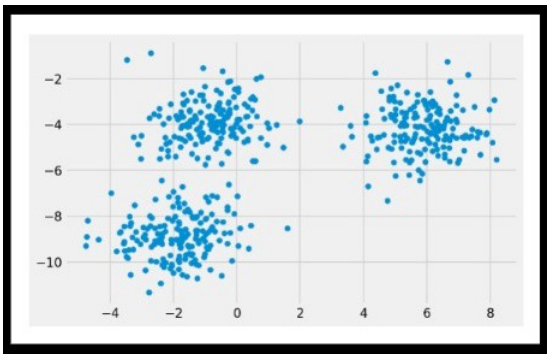

Figura 2.1: *Ejemplo de algoritmo K-Medias. Paso 1*

En la **Figura 2.1** se tiene una nube de puntos donde todavía no se ha realizado ningún agrupamiento. Vemos claramente la existencia de dos o incluso tres grupos.

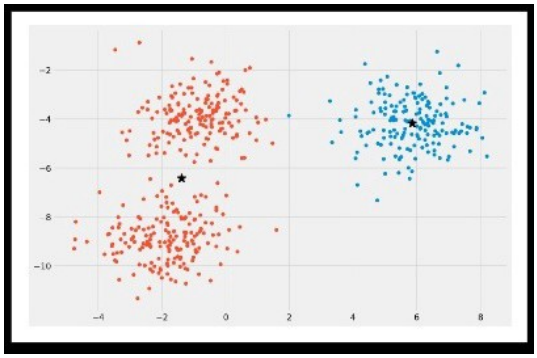

Figura 2.2: *Ejemplo de algoritmo K-Medias. Paso 2*

En la **Figura 2.2** Se ha utilizado el algoritmo K-Medias con K = 2 clústers y se observa como claramente se separan los puntos en dos grupos muy diferenciados. Pero en el grupo rojo el centroide está difuso, por lo que se podría inferir la existencia de hasta tres grupos diferentes.

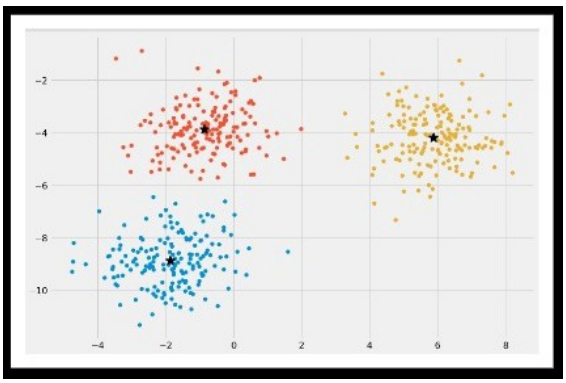

Figura 2.3: *Ejemplo de algoritmo K-Medias. Paso 3*

Utilizando el algoritmo K-Medias con K = 3 se observa como aparecen tres clústers bien definidos, además como se había supuesto con anterioridad, en el clúster rojo de la **Figura 2.2** ahora se han formado dos clústers claramente diferenciados.

Se proporciona el pseudocódigo del algoritmo con el fin de mejorar su comprensión.

K-Means:

COMIENZO
Inicializamos aleatoriamente las K-particiones.
Calcular la matriz de medias $M = [m_1, ..., m_K]$;
while $\exists$ cambios en M **do**
  clasificar objetos de acuerdo al m_i más cercano;
  recalcular M;
end while
return : M
FIN

Pseudocódigo extraído de (Gallardo Campos 2009)

Algoritmo *Fuzzy K-Means*

En el algoritmo K-Means los individuos serán asignados a un único clúster. *Fuzzy K-Means* relaja esta condición y crea un grado de pertenencia $\mu_i(x_j)$ de cada objeto x_j al clúster C_i con $0 \leq \mu_i(x_j) \leq 1$, de esta manera buscará minimizar la función de coste. Se utilizará la siguiente notación:

$$J(U, M) = \sum_{i=1}^{c} \sum_{j=1}^{N} \mu_{ij}^{b} ||x_j - m_i||^2$$

- $U = [\mu_{ij}]_{cxN}$ que hace referencia a la matriz de particiones borrosa.
- $M = [m_i, ..., m_c]$ que es la matriz de prototipos de grupos.
- b un parámetro de elección libre escogido para ajustar la mezcla de los distintos puntos.

Las probabilidades de pertenencia de cada individuo cada uno de los grupos se normalizarán según la ecuación:

$$\sum_{i=1}^{c} \hat{P}(C_i|x_j) = 1, j = 1, ..N.$$

El mínimo de la función de coste se obtiene cuando:

$$\frac{\partial J}{\partial m_i} = 0$$

$$\frac{\partial J}{\partial \hat{P}_j} = 0$$

Es decir, cuando los centros de los grupos, m_j, se encuentran cerca de aquellos individuos cuya probabilidad de pertenencia al grupo C_j es elevada. Las medias de los grupos y las probabilidades de pertenencia de los individuos se estiman iterativamente utilizando el siguiente algoritmo: (Gallardo Campos 2009)

COMIENZO
Inicializamos K-particiones aleatoriamente.
Calcular $P(C_i|x_j)$ y $M = [m_1, ..., m_c]$;
while $\exists$ cambios en M y en $P(C_i|x_j)$ **do** :
Recalcular M;
Recalcular $P(C_i|x_j)$
end while
return M;
FIN

Pseudocódigo extraído de (Gallardo Campos 2009)

Algoritmo *K-NN (K-nearest neighbours)*

Este paradigma se fundamenta en una idea muy sencilla, se quiere clasificar un nuevo elemento, por lo tanto, este algoritmo buscará la clase o grupo más frecuente a la que pertenezcan los vecinos más cercanos de dicha observación.

La notación que se empleará será la siguiente:

		X_1	...	X_j	...	X_n	C
(x_1, c_1)	1	x_{11}	...	x_{1j}	...	x_{1n}	c_1
	⋮	⋮		⋮		⋮	⋮
(x_i, c_i)	i	x_{i1}	...	x_{ij}	...	x_{in}	c_i
	⋮	⋮		⋮		⋮	⋮
(x_N, c_N)	N	x_{N1}	...	x_{Nj}	...	x_{Nn}	c_N
x	$N+1$	$x_{N+1,1}$	...	$x_{N+1,j}$	...	$x_{N+1,n}$	?

Tabla 2.1: *Notación para el algoritmo K-NN*

- Se dispone de un fichero con N casos, cada uno de los cuales está categorizado por n variables predictoras, $X_1, ..., X_n$ y una variable a predecir que es la clase C.
- Los N casos se denotarán por:

$$(\mathbf{x}_1, c_1), ..., (\mathbf{x}_N, c_N) \text{ donde}$$
$$x_{i,1} = (x_{i,1}...x_{i,n}) \text{ para todo } i = 1, ..., N$$
$$c_i \in \{c^1, ..., c^m\} \text{ para todo } i = 1, ...N$$

$c^1, ..., c^m$ denotan los m posibles valores de la variable clase C.

- El nuevo caso a clasificar se denotará por $\mathbf{x} = (X_1, ..., x_n)$.

A continuación se presentará un pseudocódigo para el clasificador K-NN básico.

COMIENZO
Entrada: $D = \{(\mathbf{x}_1, c_1), ..., (\mathbf{x}_N, c_N)\}$
$x = (x_1, ..., x_n)$ nuevo caso a clasificar
PARA todo objeto ya clasificado (x_i, c_i)
calcular $d_i = d(\mathbf{x}_i, \mathbf{x})$
Ordenar $d_i (I = 1, ...N)$ en orden ascendente
Quedarnos con los K casos $D_{\mathbf{x}}^K$ ya clasificados más cercanos a $\mathbf{x}$
Asignar a $\mathbf{x}$ la clase más frecuente $D_{\mathbf{x}}^K$.
FIN

Pseudocódigo y texto extraído de (Moujahid et al. 2019).

Como se puede observar en el pseudocódigo de arriba, se calculan las distancias de los casos ya clasificados al nuevo caso $\mathbf{x}$. A continuación, se ordenan las distancias de manera ascendente. Entre los K casos más cercanos al nuevo caso, $\mathbf{x}$, se buscará la clase más frecuente y será clasificado en dicha clase.

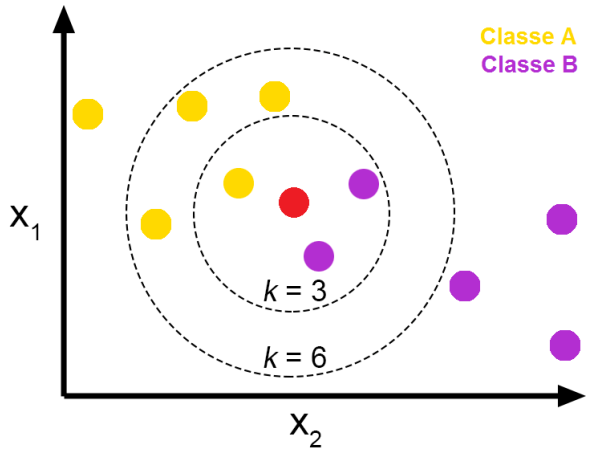

Figura 2.4: *Ejemplo de algoritmo K-NN*

En la **Figura 2.4**, se observa un punto rojo, este es el nuevo caso a clasificar. Existen dos grupos A y B, amarillo y morado respectivamente. Dependiendo del número de

vecinos (K), que se determinan al inicio del algoritmo, la nueva observación puede ser clasificada en la clase B, en el caso de que $K = 3$. Esto sucede porque dos de los tres vecinos más cercanos a esta observación pertenecen a dicha clase. Por el contrario, si $K = 6$, cuatro de los seis vecinos más cercanos pertenecen a la clase A y, por tanto, se clasificaría en esta. Este ejemplo da a denotar la importancia de la elección del número de vecinos, puesto que afecta directamente a la clasificación.

2.1. Distancias entre cadenas de texto

Este proyecto se centrará en el cálculo de similitudes de texto. En consecuencia, se definirán algunas métricas que se utilizan comúnmente en este campo. Las métricas serán divididas en dos grupos; métricas basadas en operaciones de edición y métricas de conjuntos.

2.1.1. Métricas basadas en operaciones de edición

Anteriormente se ha definido el concepto de distancia. En el marco de las comparacion de cadenas de texto también se toman en consideración las operaciones de edición. Las operaciones de edición básicas serán cuatro: *inserción, supresión de un carácter, sustitución* y *transposición*, las cuales se describirán a continuación acompañadas de ejemplos para facilitar su comprensión:

Texto original	Operación	Texto final
pato	Insertar s	pasto
pasto	Eliminar t	paso
paso	Sustituir p por c	caso
caso	Transponer s y o	caos

Tabla 2.2: *Operaciones de edición, (Campos S. Diego 2019)*

Todas las métricas que se describirán a continuación se basan en el conjunto mínimo de operaciones de edición necesarias para transformar una cadena A en otra B.

- **Distancia de Hamming**
  Esta métrica es empleada frecuentemente en el ámbito de la teoría informática, criptografía y telecomunicaciones. Siendo una de las métricas más simples. Dados $\mathbf{x}$, $\mathbf{y} \in \mathcal{A}^n$, se denomina distancia de Hamming entre $\mathbf{x}$,$\mathbf{y}$ al número de coordenadas distintas que poseen. Es decir, el número de operaciones que han de realizarse para transformar la cadena $\mathbf{x}$ en la $\mathbf{y}$.(Moro et al. 2007)

$$d(\mathbf{x},\mathbf{y}) = \#\{i \mid 1 \leq i \leq n, x_i \neq y_i\}$$

- **Distancia de Levenshtein**
  Esta métrica es comúnmente utilizada en algunos campos como la corrección ortográfica automática, los sistemas de reconocimiento del habla, el análisis de ADN y en la detección de plagio. Las operaciones de edición permitidas para el cálculo de esta distancia son: *eliminación* , *inserción* y *sustitución de un carácter*, otorgándose a cada uno un costo unitario. La distancia de Levenshtein, tiene una problemática con respecto al resto. Tiende a fallar cuando se intentan identificar cadenas equivalentes que han sido demasiado "truncadas", es decir, que se hayan utilizado abreviaturas u omitido ciertas letras.(Morales et al. 2015)

- **Alineación Óptima de Cadenas (OSA)**
  Esta distancia es una ampliación de la distancia de Levenshtein. En este caso se contabiliza la *transposición* como una operación válida. Con un ejemplo esta diferencia se puede apreciar mejor. Se quiere transformar la palabra *caso* en *caos*; OSA es capaz de detectar que si realizamos una transposición de la letra *o* y la letra *s* se obtiene directamente la palabra que se busca. Por el contrario, Levenshtein primero sustituirá la *o* por la *s*, y a continuación la *s* por la *o*, en total se realizan dos operaciones, una más que utilizando la métrica anterior.

- **Damerau-Levenshtein**
  Se define como una extensión de la métrica OSA, se añade como operación válida la transposición de dos caracteres adyacentes. (Cámara et al. 2019)
  La diferencia entre estas dos métricas se aprecia mejor con un ejemplo:

$$OSA(\text{"}ca\text{"}, \text{"}abc\text{"}) = 3,\ (\text{"}ca\text{"} \rightarrow \text{"}a\text{"} \rightarrow \text{"}ab\text{"} \rightarrow \text{"}abc\text{"})$$

$$DL(\text{"}ca\text{"}, \text{"}abc\text{"}) = 2,\ (\text{"}ca\text{"} \rightarrow \text{"}ac\text{"} \rightarrow \text{"}abc\text{"})$$

2.1.2. Métricas de conjuntos

En contraposición a las anteriores métricas, estas operan sobre conjuntos. Se utilizan para la comparación de frases o párrafos completos interpretándolos como un conjunto de palabras, también pueden ser utilizadas para la comparación de palabras si se considera cada palabra como un conjunto de letras.

- **Índice de Jaccard**
  El índice de Jaccard está definido como la intersección sobre la unión de los conjuntos y es una medida de la similitud entre ambos. Es decir, es la división entre el número de elementos en común que tienen los dos conjuntos entre el

número de elementos únicos que tiene la unión de ambos conjuntos.

$$J(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$$

Por lo tanto, la distancia de Jaccard se define como:

$$d_J(A,B) = 1 - J(A,B)$$

Después de haber realizado el proceso de *tokenización*, el cual se desarrollará más adelante, se aprecia que el índice de Jaccard no tiene en cuanta la posición que ocupa el elemento, además los elementos repetidos se consideran como uno solo dentro del conjunto.(hmong 2014)

- **Similitud del coseno**
  Dentro del dominio de la clasificación de texto, es empleada para indicar el grado de similitud que existe entre dos textos. Tomando valores entre 0 y 1, donde un valor de 0 indica que no existe similitud entre los elementos y un valor de 1 indicará que los elementos comparados son idénticos.(Park et al. 2020) (Kocher and Savoy 2017). Sean A y B los vectores de dos textos t1 y t2 respectivamente, se define la similitud del coseno como:

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

2.2. Algoritmos de clasificación de textos

En esta sección se introducirán algunos paradigmas que son frecuentemente empleados en la clasificación de textos.

2.2.1. Factorización de matrices no-negativas (*NMF*)

(Esta sección se ha basado en (Benítez et al. 2014))

Este algoritmo es utilizado comúnmente en la minería de textos, más concretamente para analizar la frecuencia de aparición de palabras. Este método resulta eficaz cuando los conjuntos de datos que se quieren analizar no toman valores negativos. Se dispondrá de un conjunto de N textos que se quieren clasificar mediante un subconjunto de K palabras. El sistema se puede representar mediante una matriz de ocurrencias, que denominaremos A, cuyo tamaño es $N \times K$. Los componentes i, j indican el número de veces que la palabra j se ha encontrado en el texto i.

$$
\begin{array}{c}
\\
\text{texto1} \\
\text{texto2} \\
\vdots \\
\text{texto N}
\end{array}
\begin{array}{c}
\begin{array}{cccc}
\text{palabra 1} & \text{palabra 2} & \dots & \text{palabra K}
\end{array} \\
\begin{pmatrix}
a_{1,1} & a_{2,1} & \dots & a_{1,k} \\
a_{2,1} & a_{2,1} & \dots & a_{2,k} \\
\vdots & \vdots & \ddots & \vdots \\
a_{N,1} & a_{N,2} & \dots & a_{N,K}
\end{pmatrix}
\end{array}
$$

Si se eligen las K palabras de la manera correcta, obtendremos como resultado una clasificación de los textos según la frecuencia de aparición de cada palabra. Por ejemplo, si la procedencia de nuestros textos en de una página web de noticias. Se podría pensar que aquellos textos en los que aparezcan frecuentemente palabras relacionadas con ayudas que proporciona el estado se podrían pensar que pertenecen al ámbito local. Sin embargo tambien existen las ayudas internacionales que se podrían llegar a mal clasificar. El método NMF salva esta ambigüedad, puesto que define un conjunto de características que van a permitir agrupar los textos de una manera muy eficiente. En resumen, NMF permite agrupar textos siguiendo un criterio que los identifica mediante el uso de palabras clave. NMF realizará una descomposición de matrices en factores. Se dispondrá de una matriz de datos no-negativa denominada A de la forma:

$$A \approx W \cdot F$$

La matriz F es la *matriz de caracteres*, de tamaño N x K, define M características y pondera cada una de ella mediante la importancia que tiene cada una de las K palabras. La componente i, j de F representa la importancia de la palabra j en la característica i.

$$
\begin{array}{c}
\\
\text{característica 1} \\
\text{característica 2} \\
\vdots \\
\text{característica M}
\end{array}
\begin{array}{c}
\begin{array}{cccc}
\text{palabra 1} & \text{palabra 2} & \dots & \text{palabra K}
\end{array} \\
\begin{pmatrix}
f_{1,1} & f_{2,1} & \dots & f_{1,k} \\
f_{2,1} & f_{2,1} & \dots & a_{2,k} \\
\vdots & \vdots & \ddots & \vdots \\
f_{M,1} & f_{M,2} & \dots & a_{M,K}
\end{pmatrix}
\end{array}
$$

Por otra parte, se tendrá la matriz W que se denominará como *matriz de pesos*, la cual, como su propio nombre indica, atribuirá un peso a cada una de las características en función de su importancia. Es una matriz N x M, donde las filas serán cada uno de los N textos y las columnas corresponderán a las características, de esta

forma la componente i, j de W indicará la importancia que tiene la característica j en el texto i.

$$
\begin{array}{c}
 \\
\text{texto 1} \\
\text{texto 2} \\
\vdots \\
\text{texto N}
\end{array}
\begin{array}{c}
\begin{array}{cccc}
\text{característica 1} & \text{característica 2} & \dots & \text{característica M}
\end{array} \\
\begin{pmatrix}
w_{1,1} & w_{2,1} & \dots & w_{1,M} \\
w_{2,1} & w_{2,1} & \dots & w_{2,M} \\
\vdots & \vdots & \ddots & \vdots \\
w_{M,1} & w_{M,2} & \dots & w_{N,M}
\end{pmatrix}
\end{array}
$$

Un valor demasiado alto resultaría en un etiquetado muy específico, lo cual produciría sobreajuste. Mientras que un valor excesivamente pequeño agruparía los textos en unas pocas características demasiado genéricas y, por tanto, carentes de información relevante. Conviene precisar que la descomposición NMF no es única, es decir, dado un número de características M hay diversas formas de descomponer los datos. El algoritmo que se encargará de realizar esta factorización sigue un proceso de optimización multidimensional que minimiza el residuo R de la descomposición.

$$R = W \cdot F = A$$

2.2.2. Word Embeddings

El Word Embeddings es una técnica de NLP que consiste en asignar un vector a cada palabra. Este vector será el que guarde la información semántica, lo cual permite que sea asociado o disociado a otros vectores (palabras) según distintos contextos gramaticales. Word Embeddings propone una solución efectiva, para codificar tanto la semántica como la relación de las palabras entre sí. La codificación es generalizable, es decir, puede ser utilizada para resolver otros problemas como la traducción de textos, entre otros (ENZYME Advising Group 16 Agosto 2019). Se describirá uno de los algoritmos más utilizados en el Word Embeddings para el análisis de similitudes entre palabras.

Word2Vec

Word2Vector se apoya en una red neuronal de dos capas para procesar datos. Como datos de entrada se utilizara el cuerpo de un texto. Como salida se obtendrán un conjunto de vectores numéricos. Este conjunto de vectores numéricos son interpretables para cualquier algoritmo que se quiere emplear. Es decir, utilizando previamente este algoritmo se podría utilizar un algoritmo de clustering, como alguno de los mencionados anteriormente. Su eficiencia viene dada por su gran capacidad para agrupar

vectores de palabras similares. En el caso de disponer de un conjunto de datos lo suficientemente extenso, este algoritmo sería capaz de proporcionar estimaciones sólidas sobre el significado de las palabras en función de su frecuencia de aparición en el texto (Abujar et al. 2019) . Seguidamente se expondrá un ejemplo del funcionamiento de Word2Vector:

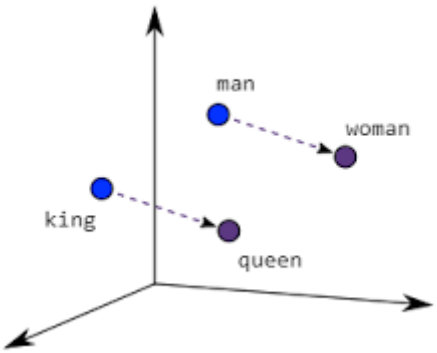

Figura 2.5: *Ejemplo del funcionamiento de Word2vec*

Al realizar operaciones algebraicas en Word Embeddings, se puede encontrar una aproximación cercana de similitudes de palabras. Como se puede apreciar en la figura de arriba, el vector *king* esta cerca del vector *man* y también relativamente próximo al vector de *women*.

Capítulo 3

Sistemas de recomendación y Procesamiento del Lenguaje Natural

3.1. Sistemas de recomendación

Los sistemas de recomendación surgieron en la década de los 90. Su finalidad era poder seleccionar los ítems más relevantes para cada persona dentro de la gran cantidad de información que se disponía en internet. Aparte de solucionar el problema de la selección de información, también surgió una oportunidad de negocio, generando beneficios con los datos recogidos por los sistemas de recomendación. En 1992, surge el primer sistema de recomendación denominado *Information Tapestry*, cuyo propósito era adaptar la entrega y la presentación de información contenida en internet y personalizarla para cada usuario. Esto favoreció la gestión de una gran cantidad de correos electrónicos, documentos, noticias por cable y artículos de NetNews. Este sistema se valía de las valoraciones y del contenido para adaptar la información a cada usuario (Terry 1993). Más tarde, en 1996, GroupLens desarrollo un sistema de recomendación denominado *MovieLens*. La finalidad de este era recomendar películas a los usuarios basándose en la información proporcionada por los mismos. Este sistema resolvía el problema del índice frío, este problema afecta directamente a los sistemas de recomendación en el que hay que realizar un modelado de datos automático. Esto se refiere a que el sistema es incapaz de efectuar inferencias para los usuarios de los cuales no posee información suficiente. En mayo de ese mismo año, GroupLens fundó la compañía NetPerceptions, la cual suministraba servicios a plataformas como E! Online y Amazon.com. Este último utilizó la tecnología de esta compañía para desarrollar su motor de recomendación en sus inicios (MovieLens Mayo 1996) , (Universidad Católica de Chile 2017). En 2006 se anunció el Premio Netflix, una competición de Machine Learning y Data Mining. La compañía ofrecía

un millón de dólares a la persona que pudiera mejorar en un 10 % la precisión de *Cinematch*, su actual sistema de recomendación. Evan Miller en su artículo de 2009 (Evan Miller 6 Febrero 2009) propone una solución bastante interesante y explica el porqué no hay que usar el promedio de las valoraciones. A mayores expone porque algunas de las soluciones que a primera vista pueden parecer correctas presentan una serie de errores en su desarrollo. En la actualidad los sistemas de recomendación son empleados por todos nosotros en nuestro día a día. Gracias al avance imparable de la Inteligencia Artificial y del Machine Learning que ha ayudado a mejorar la precisión de los paradigmas utilizados de manera exponencial.

3.1.1. Tipos de sistemas de recomendación

Primeramente antes de especificar los distintos sistemas de recomendación vamos a definirlos; Estos sistemas se encargan de filtrar la información, la cual presenta distintos tipos de temas o ítems de información como pueden ser películas, series, objetos..., que pueden llegar a ser de interés para el usuario.(Wikipedia 2022) Teniendo esto en cuenta los sistemas de recomendación son clasificados en los siguientes grupos:

Sistemas de recomendación basados en filtrados colaborativos

Un gran problema de los sistemas de recomendación es como hacer frente a la gran cantidad de información de la que se dispone, normalmente solo unos pocos datos resultan de interés para el usuario por eso a la hora de filtrar hay que intentar evitar lo máximo posible las pérdidas de tiempo que pueden desencadenar experiencias negativas para el usuario. Para estos casos, el filtrado colaborativo selecciona exclusivamente la información necesaria de cada usuario y lo compara con diversos perfiles buscando similitudes, recomendando los ítems que han resultado de interés para perfiles similares. De esta forma optimizamos el funcionamiento del recomendador. El mayor inconveniente de este tipo de filtrado es que precisa de una gran cantidad y diversidad de perfiles de usuarios para poder compararlos.(Benítez et al. 2014)
Dentro de estos sistemas se dispone de dos tipos de algoritmos:

- **Algoritmos de filtrado colaborativo basados en memoria, o algoritmos de vecinos cercanos(Nearest Neighbour)**
  En este caso se utiliza toda la base de datos, tanto de elementos como de usuarios, con el fin de generar las recomendaciones al nuevo usuario. Se suele emplear métodos estadísticos para encontrar los perfiles más similares. Uno de los algoritmos más usados es el de *K-Nearest Neighbour*. Este algoritmo, que ha sido descrito en el capítulo anterior, calculara primeramente las similitudes entre los perfiles de usuarios y realizara una lista de los N perfiles más similares

para finalmente recomendar al nuevo usuario los ítems que más han satisfecho a los perfiles más similares al suyo.(Nieto 2007)

- **Algoritmos de filtrado colaborativo basados en Modelo.**
  Primeramente se desarrolla un modelo de valoraciones del usuario, es decir, se elabora una lista con las *"valoraciones"* que le ha dado ese usuario a cada ítem concreto. Posteriormente, se calcula el valor esperado para cada ítem en función de las valoraciones otorgadas. Para realizar este proceso se emplean distintos algoritmos de Análisis Clúster o Redes Neuronales con el fin de resolver este problema de predicción. Por lo general este tipo de filtrado colaborativo es más rápido, pero a su vez precisa de una gran cantidad de datos para realizar un proceso de aprendizaje efectivo.(Nieto 2007)

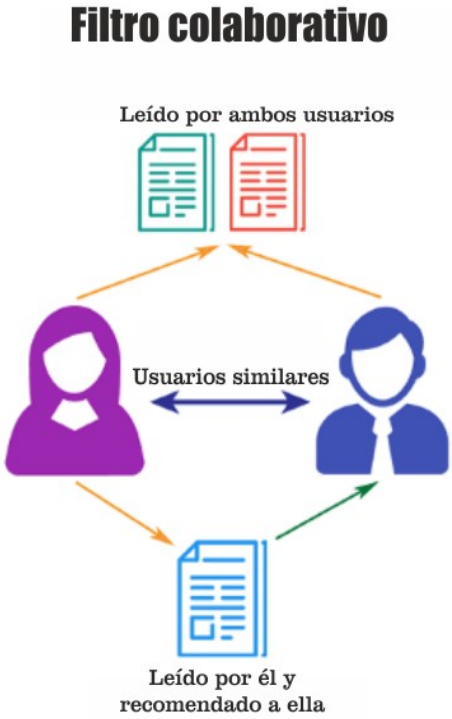

Figura 3.1: *Filtrado colaborativo*

Sistemas de recomendación basados en el contenido

En este tipo de sistemas, los ítems que han sido vistos por el usuario y a su vez los ha calificado, son usados para elaborar las nuevas recomendaciones, estos datos se guardan en un pseudo-perfil del usuario, y es este pseudo-perfil el que se emplea para encontrar los nuevos ítem que se recomendaran. Este tipo de sistemas tiene una mayor presencia en la actualidad. (de Campos et al. 2011). En el artículo,(Alvarado Garcia et al. 2015), se comenta la elaboración de un sistema de recomendación basado en

el contenido, con el fin de recomendar música a los usuarios. En este artículo señala que *Ün elemento clave en este tipo de sistemas es la adecuada representación de los contenidos, así como la existencia y veracidad de un perfil del usuario, ambos aspectos impactarán de manera directa en la efectividad de las recomendaciones hechas por el sistema."*, es decir que es importante que los datos de los que disponemos tanto como de los perfiles como de las canciones sean verídicas y a su vez que los datos estén bien estructurados.

Figura 3.2: *Filtrado basado en el contenido*

Sistemas híbridos:

Un sistema de recomendación híbrido es aquel que combina diferentes enfoques o técnicas de recomendación para producir una mejora en sus características y por consiguiente aumentar la mejora de las recomendaciones.(Burke 2007). Para la integración de diversos enfoques se disponen de diversos métodos, uno de ellos el de la ponderación consiste en combinar las valoraciones para producir una única recomendación, es uno de los sistemas más sencillos. En el siguiente artículo; (Vera et al. 2015), se describen alguno de los métodos más utilizados para combinar diversos enfoques.

3.2. Procesamiento del Lenguaje Natural

Al estudiar los problemas derivados de la generación y comprensión automática del lenguaje natural nace en la década de 1950 el Procesamiento del Lenguaje Natural (NLP). Esta técnica combina dos campos que a primera vista no están muy relacionados, la Inteligencia Artificial y la Lingüística. El NLP permite establecer una comunicación entre la máquina y el ser humano. Su origen nos lleva a la Segunda Guerra Mundial, donde surgió la necesidad de traducir frases del ruso al inglés y debido a la falta de traductores surgió la idea a de automatizar este proceso. La universidad de Georgetown junto con IBM consiguieron, en 1954, traducir automáticamente más de sesenta oraciones rusas. Durante la década de 1970, se empezaron a escribir sistemas conceptuales que permitían estructurar la información y transformar los datos para que fueran comprensibles para una máquina. Para la siguiente década, en 1980, la mayoría de los sistemas que implementaban NLP se basaban en un conjunto de reglas bastante complejas y escritas a mano. El aumento computacional que se produjo a finales de esta década propicio una revolución en el ámbito del NLP, introduciéndose de algoritmos de aprendizaje automático. Entre 1990 y los 2000, sé continuo en la misma línea que a finales de la década pasada, con la introducción de algoritmos de aprendizaje automático se pudo introducir reglas similares a las escritas a mano, permitiendo de esta manera ahorrar un gran cantidad de tiempo y de líneas de código. Al inicio del nuevo milenio, comenzó el auge de las redes neuronales, que se implementaron también para este campo. En 2001, Yoshio Bengio y su equipo propusieron el primer modelo de lenguaje basado en una red neuronal, utilizando una red *FeedForward* (Khyani et al. 2020). En la década de 2010, el aprendizaje por representación y los métodos de aprendizaje automático del estilo de las redes neuronal se generalizaron en el procesamiento del lenguaje natural, esto se dio gracias la presentación de una serie de resultados que muestran que dichas técnicas pueden lograr buenos resultados en muchas tareas del lenguaje natural como por ejemplo podría ser responder a una pregunta de manera precisa.

3.2.1. Fases del Procesamiento del Lenguaje Natural

A la hora de aplicar métodos de NLP será necesario, en la mayoría de los casos, realizar las siguientes fases que se describirán a continuación: normalización, tokenización y lematización.

Normalización

El lenguaje natural, como herramienta del ser humano, tiende a tener una naturaleza aleatoria debido a la aleatoriedad de su creador, esto dificulta el análisis del texto, por lo tanto, con la normalización lo que se busca es reducir esa aleatoriedad acercándolo a un modelo estándar. En este punto se eliminarán los acentos, los signos

de puntuación, las contracciones y las *stopwords*, también en esta fase se realizará el proceso de *Lowercased*, que consistirá en cambiar todas las letras de mayúsculas a minúsculas, con el fin de facilitar procesos futuros.

Las stopwords son aquellas palabras que carecen de sentido al ser escritas solas. Como son los artículos, las conjunciones, las preposiciones y los adverbios.

Tokenización

Se denominará tokenización al proceso mediante el cual se divide un texto es sus distintos componentes, eliminando los espacios en blanco y saltos de línea. (Nadkarni et al. 2011)

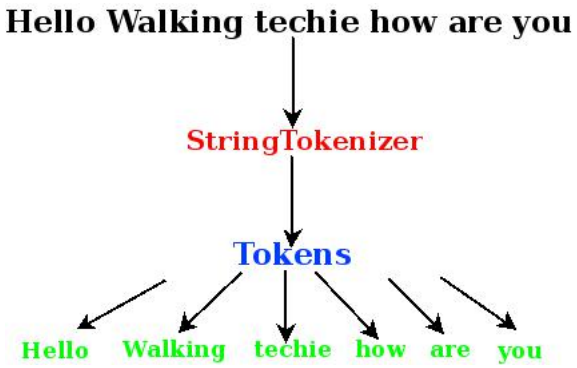

Figura 3.3: *Ejemplo de tokenización*

Como resultado se obtendrá una cadena de *tokens*. Un *token* es una serie de caracteres que tienen significado por sí mismos dentro del texto analizado.

Lematización

Para empezar a hablar de la Lematización primero se deberá definir el término *lema*, un *lema* son una serie de caracteres que conforman una unidad semántica y que puede constituir una entrada de diccionario. Básicamente es la raíz de una palabra, un ejemplo lo encontramos en la palabra *buscador* su lema es *"busc"* puesto que de ese lema derivan el resto de palabras relacionadas como *buscar, búsqueda, buscársela, buscando...*(Liddy 2001) (Kumar 2011) Este proceso se va a implementar para todas las palabras del texto a analizar. Esto nos ayudará a encontrar las keywords o palabras clave con las que se podrá identificar el texto. Las keywords son palabras que se suelen repetir constantemente en el texto y cuya importancia dentro del mismo es muy relevante. Se utilizan frecuentemente para identificar el tema de un texto o para diferenciar un texto de otros.

Una vez definido este proceso, se realizará un ejemplo donde se observará como se irá modificando una frase a medida que pasa por cada uno de los procesos descritos anteriormente:

Frase inicial	El niño corre por la calle del árbol
Normalización	'niño corre calle arbol'
Tokenización	'niño' 'corre' 'calle' ' arbol'
Lematización	'niñ' 'corr' 'calle' 'arb'

Tabla 3.1: *Ejemplo de procesos de NLP*

NLU

El Entendimiento o Comprensión del Lenguaje Natural o NLU es un subconjunto del Procesamiento del Lenguaje Natural (NLP). El objetivo principal de los paradigmas de comprensión del lenguaje natural es transformar el lenguaje humano hablado o escrito en un lenguaje que pueda ser comprensible por una máquina (Rus et al. 2009). Las técnicas de NLU se sustentan en las técnicas de NLP y a mayores integran una serié de técnicas propias, un claro ejemplo es el análisis de sentimientos donde se utiliza para determinar las emociones o la polaridad de los interlocutores. También es comúnmente empleado en la traducción de textos. Cuyo principal problema es la existencia de sinónimos en los idiomas, lo cual implica que la máquina debe percatarse del contexto primeramente para, posteriormente, poder traducirla apropiadamente.
Por último se cuenta con la Inferencia del Lenguaje Natural. Consiste en emplear redes neuronales que infieran resultados. No utiliza los clasificadores convencionales del NLP. Cabe decir que esta técnica es bastante novedosa y tiene unos resultados aceptables.

NLG

El NLG o *Generación del Lenguaje Natural* en español, es otra sub-área del NLP, en la cual se emplea la Inteligencia Artificial para generar narrativa tanto escrita como hablada a partir de un conjunto de datos. El NLG implementa tanto las técnicas de *Procesamiento del Lenguaje Natural* como las del *Entendimiento del Lenguaje Natural* y las áreas de la Inteligencia Artificial que involucran la comunicación entre los humanos y las máquinas y viceversa. El NLG precisa de una gran cantidad de datos, hay que tener en cuenta que para que esto se pueda llevar a cabo hay que realizar un proceso de aprendizaje previo, donde la máquina aprenderá el vocabulario, las expresiones y la sintaxis que constituye una lengua, lo cual presenta un gran problema de almacenamiento. Un claro ejemplo son los chatbots, asistentes virtuales

que sobre todo se han implementado en las empresas y que no siempre son capaces de resolver nuestras dudas.

Comparación entre NLP y NLU

Si se compara el NLP con el NLU se observa que la mayor diferencia que presentan es que NLU tiene en cuenta la subjetividad del lenguaje, como puede ser los tipos de palabras que se usan o el tono en el que se pronuncian, por otra aparte NLP tiene más relación con los procedimientos intrínsecos como el etiquetado gramatical. En el siguiente gráfico se aprecian los distintos campos a los que pertenecen diversos procedimientos del procesamiento del lenguaje.

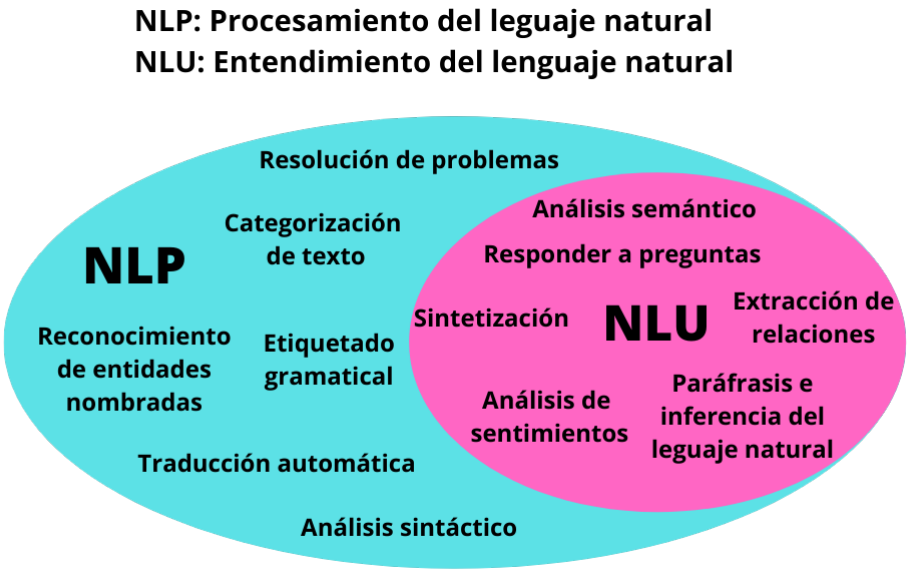

Figura 3.4: *NLP vs NLU*

NLP en la actualidad

Anteriormente se ha comentado como se ha ido desarrollando el NLP a lo largo de la historia, pero en la actualidad ¿dónde se pueden observar esas implementaciones? A continuación se describirán algunos ámbitos donde el NLP tiene mucha importancia.

1. **Google search.** Un ejemplo es el motor de búsqueda más utilizado en el mundo, el de Google, actualizo el algoritmo de búsqueda e implemento **BERT** que es el acrónimo para ***Bidirectional Encoder Representations from Transformers***. Se trata de un sistema basado en Inteligencia Artificial que ayuda a los algoritmos de Google Search a comprender mejor el lenguaje empleado por los usuarios. Es decir, **BERT** pone en contexto la consulta realizada. Esto lo consigue, puesto que posee una característica denominada *'bidireccionalidad'*

que consiste en analizar una oración en dos direcciones, analiza las palabras que se encuentran a ambos lados de una de las keywords permitiéndole de esta manera profundizar en el contexto y la temática de la frase completa que introduce el usuario.

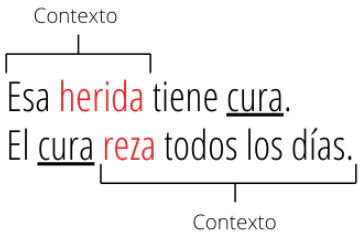

Figura 3.5: *Bidireccionalidad de BERT*

Como se puede apreciar en la **Figura 4.3** se tiene la palabra clave *cura*, al analizar en ambas direcciones las dos frases se aprecia el contexto, donde en la primera oración la palabra cura hace referencia al proceso de sanar una herida y en la segunda se refiere a un sacerdote de la iglesia católica.

2. **Decisiones financieras.** Otra aplicación poco conocida del NLP consiste en extraer información relevante y trascendental con el fin de mejorar las decisiones comerciales. Últimamente, muchas de las decisiones importantes que se toman en los mercados financieros ya no pasan por las manos del hombre. La negociación algorítmica es cada vez más popular, este nuevo método consiste en la programación de decisiones de trading[1] para automatizar instrucciones de mercado. El problema es que muchas de estas decisiones financieras se ven afectadas por las conocidas como *fake news*, en estos casos los procedimientos del NLP pueden ayudar a extraer información y transformarlos a un formato el cual se pueda incluir en las decisiones de negociación algorítmica.

3. **Análisis de sentimientos.** En el artículo (Dubiau and Ale 2013), se define el Análisis de Sentimientos como: "*el estudio computacional de opiniones, sentimientos y emociones expresadas en textos*". El objetivo principal es determinar que emoción o actitud tiene el individuo frente a determinadas situaciones, productos... Para realizar correctamente este análisis se debe de realizar un preprocesamiento del texto, utilizando las técnicas mencionadas anteriormente. A continuación se procederá a la extracción de palabras clave y se extrae la oración donde se encuentra dicha palabra y se calcula la polaridad del texto conforme a una métrica previamente determinada. En el trabajo (Ballesteros Diez 2017) se emplea Cognitive Service, que es una herramienta diseñada por Microsoft que

[1]El trading consiste en la especulación sobre instrumentos financieros con el objetivo de obtener un beneficio.

permite efectuar un análisis de sentimientos a partir de imágenes de personas. Twitter es una red social donde los usuarios suelen expresar opiniones sobre diversos temas, esto hace que los textos que se pueden extraer de los tweets sean idóneos para realizar este tipo de análisis. En el artículo (SAURA et al. 2018) se efectúa un análisis de sentimientos de los tweets sobre las ofertas del Black Friday. Se filtran los tweets por el nombre de la compañía y a continuación se evalúa su polaridad, clasificándolos en *Negativos*, *Neutrales* y *Positivos*. Esta técnica es comúnmente empleada por los profesionales del marketing, con el fin de extraer la información de millones de mensajes en redes sociales o en encuestas online con el que determinar la polaridad de un servicio o producto y desarrollar mejores planes de marketing.

4. **Atención al cliente.** En los últimos años la mayoría de las empresas han implementado chatbots, un asistente que se comunican con los usuarios a través de mensajes de texto, con el fin de mejorar y ampliar el horario de atención al cliente. Los chatbots emplean tanto NLP como NLU, y otra sub-área del NLP denominada NLG, combinando estos dos procedimientos se desarrolla un chatbot. El NLP se emplea para dividir la entrada del usuario en oraciones y palabras, se transforma el texto para que nos resulte más fácil de procesar. Por su parte, el NLU ayuda al asistente a comprender lo que el usuario necesita o pregunta proporcionándole un contexto. El NLG desarrolla la respuesta a dicha consulta, al tener capacidad de aprender y consultar repositorios de datos, las respuestas proporcionadas no son prefabricadas sino personalizadas. En el año 2011, Apple dio a conocer a Siri, uno de los primeros asistentes que combinaban NLP e Inteligencia Artificial que podía ser empleado por todos los consumidores. Dentro de Siri, el módulo de reconocimiento de voz automatizado traduce las palabras del propietario en conceptos interpretables por una computadora. El sistema de comandos de voz luego hace coincidir esos conceptos con comandos predefinidos e inicia posteriormente acciones específicas.(Deep Talk 29 Noviembre 2021)

Capítulo 4

Aplicación con datos de CVs

En este capítulo se comentarán los métodos utilizados para desarrollar este proyecto. Se especificarán los datos empleados, las librerías utilizadas y el código desarrollado. Los datos han sido proporcionados por Air Instituted, lo cual hace de este proyecto un proyecto de transferencia con dicha empresa.

4.1. Datos empleados

Los datos en un primer momento, fueron extraídos en forma de enlaces web, texto o imágenes. En la parte de ingesta se han empleado herramientas actuales de programación conocidas como *web scrapinbg* y *web crawling* que permiten mandar peticiones a diferentes páginas webs recogiendo su código HTML y extrayendo su contenido. Tanto los datos de los perfiles como los de las ofertas de empleo han sido proporcionados en formato *JSON*, este es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript.

4.1.1. Datos de los perfiles laborales

El conjunto de datos de los perfiles laborales constaba de las siguientes variables:

- `name`: Nombre de la persona a la que pertenece el perfil. (*String*)
- `email`: Correo electrónico. (*String*)
- `description`: Breve descripción del perfil. (*String*)
- `links`: Link del perfil. (*String*)
- `awards`: Premios o competiciones en las que ha participado. (*String*)
- `languages`: Idiomas y nivel que posee el perfil. (*String*)

- `certifications`: Certificados que se poseen. (*String*)
- `experience_time`: Experiencia laboral hasta la fecha. (*Fecha*)
- `keywords`: Palabras clave que identifican el perfil. (*String*)
- `company_jobs`: Empresas en las que ha trabajado. (*String*)
- `jobs_date`: Fechas en las que se trabajo en dichas empresas. (*Fecha*)
- `education`: Centros de estudios donde se ha formado. (*String*)
- `jobs_duration`:Duración de los distintos trabajos. (*Fecha*)

4.1.2. Datos de las ofertas laborales

Por otro lado, el conjunto de datos de ofertas laborales disponía de las siguientes variables:

- `indice`: Nos indica el número de la oferta, cada oferta tiene un índice único. (*Integer*)
- `category`: Categoría en la cual han sido clasificada la empresa. (*String*)
- `source_page`: Página de la cual se ha extraído dicha la oferta. (*String*)
- `hash`: Campo utilizado para no repetir la oferta. (*String*)
- `title`: Encabezado o título de la oferta.(*String*)
- `organization`: Empresa que oferta el puesto. (*String*)
- `url`: URL de la página web de la que se extrajo la oferta. (*String*)
- `logo`: Logo de la página web de donde se extrajo la oferta. (*String*)
- `location`: Lugar físico donde se encuentra la empresa. (*String*)
- `requirements`: Requerimientos de la oferta. (*String*)
  - `minimun`: Número mínimo de años de experiencia. (*String*)
  - `languages`: Idioma. (*String*)
  - `education`: Nivel de educación requerido. (*String*)
  - `skills`: Habilidades requeridas para el puesto. (*String*)
- `begin_date`: Fecha de inicio del contrato. (*Fecha*)
- `salary`: Salario del puesto. (*Integer*)

- `work_position`: Posición requerida. (*String*)
- `workday`: Jornada laboral.(*String*)
- `description`: Breve descripción de la oferta. (*String*)
- `keywords`: Palabras clave que describen la oferta. (*String*)
- `others`: Otros datos de interés como el número de inscritos y de vacantes. (*String*)

4.1.3. Datos de los cursos

El último conjunto de datos utilizado se creó utilizando un diccionario de Python. Estos diccionarios emplean un sistema de clave-valor, cuyas claves, en el caso que compete al proyecto, serán las diversas habilidades que pueden llegar a solicitar en una oferta de empleo. Designaremos las distintas URL de los cursos recomendados en el campo de valor. Relacionando de esta manera cada habilidad con una formación específica de dicho campo.

4.2. Librerías utilizadas

En el transcurso del desarrollo de este paradigma se han empleado distintas librerías de `Python`. Seguidamente se proporcionara una breve descripción de su funcionamiento. La información proporcionada a continuación, ha sido extraída de la documentación de dichas librerías.

- `Librería NumPy`
  NumPy es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente cuando se dispone de un gran volumen de datos. Incorpora una nueva clase de objetos llamados arrays, permitiendo representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación.

- `Librería Pandas`
  Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos. Define nuevas estructuras de datos basadas en los arrays de la librería NumPy, pero con nuevas funcionalidades. Permite leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL. Permite acceder a los datos mediante índices o nombres para filas y columnas. Ofrece métodos para reordenar, dividir y combinar conjuntos de datos. Permite trabajar con series temporales. Realiza todas estas operaciones de manera muy eficiente.

- `Librería json:`
  Este módulo de Python se utiliza para leer y transformar los JSON en Dataframes principalmente. También se puede emplear para transformar DataFrames en JSON.

- `Librería re:`
  Este módulo proporciona operaciones de coincidencia de expresiones regulares

- `Librería codecs:`
  Este módulo define clases base para codecs Python estándar (codificadores y decodificadores) y proporciona acceso al registro interno de codecs Python, que gestiona el proceso de búsqueda de codec y manejo de errores.

- `Librería spaCy:`
  Es una biblioteca de Procesamiento del Lenguaje Natural Python diseñada específicamente con el objetivo de ser una biblioteca útil para implementar sistemas listos para producción. Es particularmente rápido e intuitivo, por lo que es un competidor superior para las tareas de Procesamiento del Lenguaje Natural (NLP).

4.3. Código empleado

Con el fin de facilitar la comprensión del algoritmo se ha desarrollado un pseudocódigo, en el cual se describirá como se implementan las distintas clases y su funcionamiento tanto individual como en conjunto.

ALGORITMO 1 Recomendador de ofertas laborales

Input: *json* jobs, *integer* job_number, *json* profiles_cv, *json* courses

```
import ReadCV
import ReadJobs
import Similarity
import CreateProfile
import Recommender

class CVRecommender:

function RECOMMENDER(jobs, job_number, profiles_cv, courses)

    jobs = ReadCV.read_jobs(jobs)
    profiles = ReadJobs.read_cvs(profiles_cv)
    profiles_bag_of_words = CreateProfile.getprofile(profiles)

    profile_created = CreateProfile.create_profile(profiles_bag_of_words)

    offert = jobs [total][job_number]

    data = Similarity.cosine_similarity(offert,profile_created)

    data_final = Recommender.course_recommender(data, courses)

    return data_candidates, data_admitted, data_noneadmitted

end function
```

Como se puede apreciar en el pseudocódigo la clase *CVRecommender* precisa de cuatro valores de entrada; el conjunto de datos de los perfiles, las ofertas, los cursos y a mayores se exige un *Integer* haciendo referencia al número de la oferta. Esto último se debe a que las ofertas laborales se encuentran en un mismo conjunto de datos y existe la necesidad de identificarlos mediante un ID.

A continuación se describirá tanto el funcionamiento como la finalidad de cada una de las clases que componen el algoritmo del recomendador.

- **Clase** `ReadJobs` :
  La finalidad de esta clase es leer los datos de las ofertas laborales y transformarlos al formato DataFrame de la librería `Pandas`. A mayores se encarga de extraer los elementos principales de cada oferta, como son las keywords y los requerimientos mínimos, así como de codificar el número mínimo de años de experiencia.

- **Clase** `Preprocess` :
  Esta clase no está implementada en el pseudocódigo de *CVRecommender*, puesto que se utiliza dentro de la `Clase ReadCV`. Tiene como finalidad transformar los `Strings` en `Integers` así como de realizar el proceso de normalización y lematización.

- **Clase** `ReadCV` :
  Su función principal será transformar el conjunto de datos de los perfiles de formato JSON a formato DataFrame, que es mucho más tratable. Los datos al haber sido extraídos mediante *web scraping* se han apreciado ciertas anomalías, como el desplazamiento de columnas o la repetición de campos. Por consiguiente, algunas columnas y filas del DataFrame no contienen datos correctos.

- **Clase** `CreateProfile` :
  Esta clase será la encargada de generar un perfil del candidato, basándose para ello, en una bolsa de palabras listada y clasificada.

- **Clase** `Similarity` :
  Se calculará la similaridad que existe entre el perfil del candidato y la oferta de empleo con esta clase. Para ello se ha decidido utilizar como métrica la similaridad del coseno, puesto que es la más adecuada para el tipo de comparaciones.

- **Clase** `Recommender` :
  En esta última parte se realiza una recomendación formativa que puede resultar de interés a los perfiles que no han sido seleccionados.

4.4. Resultados

En este apartado se expondrán los resultados obtenidos al ingresar una serie de datos de empleos y perfiles laborales al algoritmo del recomendador. Se seguirá la siguiente estructura; Primeramente se mostrarán los perfiles de los candidatos, cabe decir que se han empleado nombres codificados con el fin de salvaguardar la privacidad y la confidencialidad de los individuos a los que pertenecen estos datos. También se debe aclarar que aunque el nombre de los perfiles puedan coincidir en varios ejemplos, no hacen referencia al mismo individuo. A continuación, se mostrara la oferta de empleo a la que los candidatos han aplicado. Finalmente se expondrán los resultados obtenidos tras solicitar la recomendación al algoritmo.

Se comienza describiendo los perfiles, como ya se ha mencionado anteriormente las clases *ReadCV* y *CreateProfile* leen y acomodan los datos para disponer de ellos en un formato más tratable, por lo tanto, se proporcionarán los datos como si ya hubieran sido tratados por ambas clases de preprocesamiento.

Ejemplo 1.

La tabla siguiente corresponde a los distintos candidatos:

Name	**profile_final**
Profile 1	semisenior ingenieria full español java python informatico c
Profile 2	semisenior informatico ingeniero machine learning none linux
Profile 3	semisenior ingenieria informatico español
Profile 4	español informatico senior

Tabla 4.1: *Descripción de los perfiles del* ***Ejemplo 1****.*

A continuación se describe la oferta de empleo:

indice	0
category	informatica
work_position	administrador
keywords	linux lamp apache mysql mariadb open stack
req_minimo	semisenior
req_lenguaje	ingles
req_education	None
skills	None

Tabla 4.2: *Descripción de la oferta de empleo del* ***Ejemplo 1****.*

Después de calcular la similitud entre la oferta y los candidatos, se obtiene la siguiente tabla:

Name	**Similitud**
Profile 1	0.072441
Profile 2	0.155929
Profile 3	0.098680
Profile 4	0.000000

Tabla 4.3: *Similitudes de los perfiles y la oferta* ***Ejemplo 1****.*

En la Tabla 4.3 se observa el resultado de calcular la similitud del coseno de cada candidato con la oferta de empleo que indicada anteriormente. Observamos que el perfil que más símil con la oferta de empleo es el **Profile 2**, en el ámbito de la comparación de textos es complicado obtener valores altos de similitud, puesto que las métricas utilizadas se basan en la comparación de conjuntos. Una vez obtenida la similitud, se precederá a clasificar a los candidatos en tres grupos, los candidatos cuya similitud con la oferta es próxima a cero serán descartados:

Name	**Similitud**
Profile 4	0.000000

Tabla 4.4: *Candidatos descartados.* ***Ejemplo 1****.*

El siguiente grupo corresponde a los candidatos que no cumplen todos los requerimientos para ser admitidos, pero que su similitud es mayor que cero

Name	**Cursos Recomendados**
Profile 1	https://es.coursera.org/search?query=mysql, https://www.educaweb.com/nf/cursos-de/mysql/
Profile 2	https://es.coursera.org/search?query=mysql, https://www.educaweb.com/nf/cursos-de/mysql/
Profile 3	https://es.coursera.org/search?query=mysql, https://www.educaweb.com/nf/cursos-de/mysql/

Tabla 4.5: *Candidatos a los que se le recomiendan cursos.* ***Ejemplo 1****.*

En el último grupo estarían los candidatos que si se podrían considerar para el puesto, en este caso, no se han obtenido candidatos lo suficientemente cualificados para desempeñar dicha función por lo que no se recomienda ningún perfil.

Ejemplo 2:

A continuación pondremos otro ejemplo distinto. Seguirá la estructura determinada anteriormente.

Los candidatos que se han presentado a esta oferta son los siguientes:

Name	profile_final
Profile 1	informatico español senior
Profile 2	python semisenior c full español ingenieria java informatico
Profile 3	machine semisenior ingeniero learning informatico linux
Profile 4	junior

Tabla 4.6: *Descripción de los perfiles del* ***Ejemplo 2.***

La oferta a la que se han inscrito los distintos candidatos:

indice	4
category	informatica
work_position	programador
keywords	pl/sql sql
req_minimo	junior
req_lenguaje	ingles
req_education	None
skills	None

Tabla 4.7: *Descripción de la oferta de empleo del* ***Ejemplo 2.***

Se calcula la similitud entre la oferta y los candidatos:

Name	Similitud
Profile 1	0.0
Profile 2	0.0
Profile 3	0.0
Profile 4	0.303216

Tabla 4.8: *Candidatos descartados.* ***Ejemplo 2.***

En este caso el más apto para el puesto es el **Profile 4**, que curiosamente la única característica que posee es que tiene un perfil junior. Lo más lógico sería pensar que los candidatos más fuertes, con una experiencia mayor y más habilidades tuvieran más posibilidades de optar al puesto. Lo que sucede aquí es que en la oferta se busca un perfil junior, descartando de esta manera los perfiles senior o semisenior. Es por

este motivo que el que tiene mayor similitud es el **Profile 4**.
En este caso no se recomendarían cursos a ningún perfil, puesto que los otros cuatro candidatos serían descartados automáticamente. La salida proporcionada de los candidatos al puesto sería la siguiente:

Name	**Similitud**
Profile 4	0.303216

Tabla 4.9: *Posibles candidatos para el puesto.* ***Ejemplo 2.***

Ejemplo 3

En este último ejemplo vamos ver cual sería la salida, en el caso de tener los tres tipos de perfiles, es decir, perfiles que han sido seleccionados para el puesto, perfiles a los que se les recomendará cursos de formación y por último, los perfiles que han sido descartados. Se comienza describiendo los perfiles de los candidatos como en los ejemplos anteriores:

Name	**profile_final**
Profile 1	ingles junior
Profile 2	full junior
Profile 3	junior sql java
Profile 4	full senior

Tabla 4.10: *Descripción de los perfiles del* ***Ejemplo 3.***

En segunda instancia se describe la oferta a la que los candidatos han postulado:

indice	9
category	informatica
work_position	desarrollador web
keywords	java eclipse sql html css xml json jdbc webservices javascript
req_minimo	junior
req_lenguaje	None
req_education	None
skills	None

Tabla 4.11: *Descripción de la oferta de empleo del* ***Ejemplo 3.***

Antes de ver la tabla que nos indicará cual de todos los perfiles es el más idóneo para el puesto se mencionará cuál ha de ser a primera vista el resultado que se debería

obtener según los datos proporcionados por las tablas 4.10 y 4.11. En la oferta solicitan un desarrollador web que tenga unas ciertas habilidades como el conocimiento de ciertos lenguajes de programación (Java, SQL, HTML, etc). Además se requiere que el candidato tenga un perfil junior, es decir con menos de dos años de experiencia laboral. Se examinarán las habilidades que tienen los diferentes candidatos, el **Profile 4** sería descartado porque su perfil es senior. Lo siguiente es buscar entre las habilidades requeridas cu'ąles poseen los candidatos restantes. En este caso solo el **Profile 3** posee las habilidades necesarias para el puesto. Por lo tanto, sería de esperar que este perfil sea el que mayor similaridad obtenga de los cuatro y que el **Profile 4** sea el que menos.

Name	**Similitud**
Profile 1	0.127259
Profile 2	0.127259
Profile 3	0.219511
Profile 4	0.0

Tabla 4.12: *Posibles candidatos.* ***Ejemplo 3.***

Se comprueba que las previsiones que se realizarón se cumplen, como se había previsto, el candidato más idóneo es el **Profile 3** y el que queda descartado es el **Profile 4**. Ahora el siguiente paso consiste clasificar los perfiles obteniendo como resultado las siguientes tres tablas:

Name	**Similitud**
Profile 3	0.219511

Tabla 4.13: *Posible candidato para el puesto.* ***Ejemplo 3.***

Name	**Cursos Recomendados**
Profile 1	https://es.coursera.org/search?query=sql, https://www.educaweb.com/nf/cursos-de/sql/
Profile 2	https://es.coursera.org/search?query=sql, https://www.educaweb.com/nf/cursos-de/sql/

Tabla 4.14: *Candidatos a los que se le recomiendan cursos.* ***Ejemplo 3.***

Conclusiones y trabajos futuros

El propósito general de este proyecto era desarrollar una herramienta que dotara de una mayor agilidad y eficacia en el proceso de selección de personal con un fin aplicativo para cualquier departamento de recursos humanos. De manera complementaria, la herramienta proporciona a los postulantes una respuesta objetiva de su afinidad a la posición laboral, anexando una recomendación de mejora técnica formativa. En la consecución del objetivo marcado se empleó algoritmia basada en el Procesamiento del Lenguaje Natural (NLP) y en el Entendimiento del Lenguaje Natural (NLU). La métrica de valoración de la adecuación de los postulantes se fundamentó en el uso de la similaridad del coseno. En la sección de resultados se ha podido comprobar que el funcionamiento del algoritmo es correcto, seleccionado en todos los casos los perfiles con una mayor similaridad con la oferta de empleo.

El alcance de este proyecto se ha limitado a concretar el caso de uso dentro del ámbito de la informática. El siguiente paso verdría dado por ampliar el horizonte temático de este, extrapolándolo a diversos campos; sanidad, economía, agricultura, etc. A nivel metodológico se emplearían técnicas de clustering de manera previa con objeto de clasificar CVs y ofertas laborales en función del ámbito al que pertenezcan con el fin de dotar al algoritmo de una mayor versatilidad y adecuación al propósito marcado.

Bibliografía

Abujar, S., A. K. M. Masum, M. Mohibullah, S. A. Hossain, and Coauthors, 2019: An approach for bengali text summarization using word2vector. *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 1–5.

Alvarado Garcia, J. R., J. V. Hernandez Garcia, E. Villatoro Tello, A. G. Ramirez de la Rosa, and C. Sanchez Sanchez, 2015: Sistema de recomendación de música basado en aprendizaje semi-supervisado.

Ballesteros Diez, M., 2017: Big data para el análisis de sentimientos en imágenes. B.S. thesis, Universitat Politècnica de Catalunya.

Benítez, R., G. Escudero, S. Kanaan, and D. M. Rodó, 2014: *Inteligencia artificial avanzada*. Editorial UOC.

Burke, R., 2007: Hybrid web recommender systems. *The adaptive web*, 377–408.

Cámara, R. V., D. Campos-Sobrino, and M. C. Soberanis, 2019: Optimización evolutiva de contextos para la corrección fonética en sistemas de reconocimiento del habla. *Res. Comput. Sci.*, **148 (8)**, 293–306.

Campos S. Diego, 2019: `https://medium.com/@diego.campos.sobrino/m%C3%A9tricas-de-similitud-para-cadenas-de-texto-parte-i-introducci%C3%B3n-ba252fa64827`.

de Campos, L. M., J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales, 2011: Uso de conocimiento estructurado en un sistema de recomendación basado en contenido. *Departamento de Ciencias de la computacón e Inteligencia Artificial, Universidad de Granada, Granada, España*.

Deep Talk, 29 Noviembre 2021: `https://blog.deep-talk.ai/historia-y-actualidad-del-procesamiento-de-lenguaje-natural-8de41a357ca9`.

Dubiau, L., and J. M. Ale, 2013: Análisis de sentimientos sobre un corpus en español: Experimentación con un caso de estudio. *XIV Argentine Symposium on Artificial Intelligence (ASAI)-JAIIO 42 (2013)*.

ENZYME Advising Group, 16 Agosto 2019: `https://blog.enzymeadvisinggroup.com/natural-language-processing`.

Evan Miller, 6 Febrero 2009: `https://www.evanmiller.org/how-not-to-sort-by-average-rating.html`.

Gallardo Campos, M., 2009: Aplicación de técnicas de clustering para la mejora del aprendizaje. M.S. thesis.

Hinestroza Ramírez, D., 2018: El machine learning a través de los tiempos, y los aportes a la humanidad.

hmong, 2014: `https://hmong.es/wiki/Jaccard_index`.

Khyani, D., B. Siddhartha, N. Niveditha, and B. Divya, 2020: An interpretation of lemmatization and stemming in natural language processing. *Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology*, **22**, 350–357.

Kocher, M., and J. Savoy, 2017: Distance measures in author profiling. *Information processing & management*, **53 (5)**, 1103–1119.

Kumar, E., 2011: *Natural language processing*. IK International Pvt Ltd.

Liddy, E. D., 2001: Natural language processing.

Morales, A., and Coauthors, 2015: Sistema recomendador orientado a la educación basado en la distancia entre likes de facebook y conceptos. *publicado en Revista Tecnología e Innovación*, **2 (5)**, 921–928.

Moro, E. M., C. M. Gómez, and D. R. Benito, 2007: bases de gröbner: aplicaciones a la codificación algebraica. *Instituto Venezolano de Investigaciones Científicas*.

Moujahid, A., I. Inza, and P. Larranaga, 2019: Tema 5. clasificadores k-nn. *Departamento de Ciencias de la Computación e inteligencia artificial, Universidad del País Vasco-Euskal Herriko Unibertsitatea*, **3**, 1.

MovieLens, Mayo 1996: `https://movielens.org/`.

Nadkarni, P. M., L. Ohno-Machado, and W. W. Chapman, 2011: Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, **18 (5)**, 544–551.

Nieto, S. M. G., 2007: Filtrado colaborativo y sistemas de recomendación. *Inteligencia en Redes de Comunicaciones. Madrid*.

Park, K., J. S. Hong, and W. Kim, 2020: A methodology combining cosine similarity with classifier for text classification. *Applied Artificial Intelligence*, **34 (5)**, 396–411.

Peña, D., 2002: *Análisis de datos multivariantes*, Vol. 24. McGraw-hill Madrid.

Rebollo, F. F., and D. Barrojo, 2009: Aprendizaje por refuerzo. *Aprendizaje Automático, Departamento de Informática, Escuela Politécnica Superior*.

Rus, V., P. M. McCarthy, D. S. McNamara, and A. C. Graesser, 2009: Natural language understanding and assessment. *Encyclopedia of artificial intelligence*, IGI Global, 1179–1184.

SAURA, J. R., A. Reyes-Menéndez, and P. PALOS-SANCHEZ, 2018: Un análisis de sentimiento en twitter con machine learning: Identificando el sentimiento sobre las ofertas de# blackfriday. *Revista Espacios*, **39 (42)**.

Terry, D. B., 1993: A tour through tapestry. *Proceedings of the conference on Organizational computing systems*, 21–30.

Universidad Católica de Chile, 2017: `https://recommendersys.wordpress.com/`.

Vera, J., A. O. Mamani, and K. Villalba, 2015: Modelo de sistema de recomendación de objetos de aprendizaje en dispositivos móviles, caso: Desarrollo del pensamiento computacional. *XX Congreso Internacional de Informática Educativa, TISE 2015. Nuevas Ideas en Informática Educativa (Santiago, Chile, 1-3 de diciembre de 2015)*, 730–734.

Wikipedia, 2022: Sistema de recomendación — wikipedia, la enciclopedia libre. URL https://es.wikipedia.org/w/index.php?title=Sistema_de_recomendaci%C3%B3n&oldid=142185884, [Internet; descargado 16-marzo-2022].