Vigilada Mineducación

REVISIÓN DE TÉCNICAS ESTADÍSTICAS BAYESIANAS PARA LA COINCIDENCIA DE ENTIDADES EN CONJUNTOS DE DATOS GRANDES

Review Of Bayesian Statistical Techniques For Entity Matching In Large Datasets

SEBASTIAN LÓPEZ VALENCIA

Proyecto de Grado

Directora, profesora
Biviana Marcela Suarez Sierra

UNIVERSIDAD EAFIT
ESCUELA DE CIENCIAS APLICADAS E INGENIERÍA
MAESTRÍA EN CIENCIAS DE LOS DATOS Y ANALÍTICA
2024

CONTENIDO

ÍNDICE DE TABLAS, ILUSTRACIONES Y ECUACIONES

Tablas:

Ilustraciones:

Ecuaciones:

RESÚMEN

En el contexto del análisis de datos, la coincidencia de entidades es una tarea crucial que implica identificar y emparejar registros que representan la misma entidad a través de diferentes bases de datos. Este trabajo, aborda una revisión de diversas técnicas estadísticas, con un enfoque particular en los métodos bayesianos, para abordar este problema en conjuntos de datos extensos.

En el marco teórico y estado del arte, se revisan diversas técnicas de coincidencia, incluyendo métodos basados en reglas, funciones de distancia para texto, y métodos basados en aprendizaje automático. Se presentan también varias estrategias de optimización para reducir el costo computacional asociado con la coincidencia de entidades, incluyendo heurísticas, medidas de distancia menos complejas, y algoritmos de aprendizaje de rápida convergencia. Un enfoque destacado es el de agrupar y luego comparar entidades, lo cual reduce significativamente la complejidad de las comparaciones necesarias.

En la sección de descripción de los datos, se detallan los procedimientos de adquisición y preprocesamiento de datos, que son fundamentales para asegurar la calidad y relevancia de los conjuntos de datos utilizados en los experimentos. La metodología de trabajo se describe en detalle, abarcando desde el conocimiento del negocio hasta la adquisición, comprensión y modelado de los datos.

Finalmente, en el desarrollo de los métodos y resultados, se presentan los hallazgos obtenidos a través de la aplicación de las técnicas revisadas y propuestas en esta tesis. Las conclusiones destacan la efectividad de las técnicas bayesianas y sugieren áreas para futuras investigaciones.

ABSTRACT

In the context of data analysis, entity matching is a crucial task that involves identifying and pairing records that represent the same entity across different data sources. This work provides a review of various statistical techniques, with a particular focus on Bayesian methods, to address this problem in large datasets.

In the theoretical framework and state of the art, various matching techniques are reviewed, including rule-based methods, text distance functions, and machine learning-based methods. Several optimization strategies are also presented to reduce the computational cost associated with entity matching, including heuristics, less complex distance measures, and fast-converging learning algorithms. A notable approach is to group and then compare entities, which significantly reduces the complexity of the necessary comparisons.

In the data description section, the procedures for data acquisition and preprocessing are detailed, which are fundamental to ensure the quality and relevance of the datasets used in the experiments. The work methodology is described in detail, covering everything from business knowledge to data acquisition, understanding, and modeling.

Finally, in the development of methods and results, the findings obtained through the application of the reviewed and proposed techniques in this thesis are presented. The conclusions highlight the effectiveness of Bayesian techniques and suggest areas for future research.

INTRODUCCIÓN

Planteamiento del problema

Identificar que dos o más entidades hacen referencia a la misma persona, objeto, animal u otros en bases de datos heterogéneas es un problema ampliamente documentado (Brunner & Stockinger, 2020). La revolución en la forma en cómo se captura información, se procesa, se almacena y se comparte, apoyada por la computación, ha permitido que sea posible manejar muchas características relacionadas a una misma entidad y capturadas por diferentes sistemas, organizaciones o personas (Ramezani, Ilangovan, & Kum, 2021). A la hora de generar una base de conocimiento en torno a una entidad, suele ser requerido conciliar con las diferentes representaciones de dicha entidad en los sistemas fuente que suelen ser heterogéneas y mantenidos por organizaciones diferentes. Este proceso recibe nombres como: coincidencia de registros, enlace de datos, emparejamiento de datos, entre otros.

Un ejemplo de aplicación puede darse en el campo de la medicina: A lo largo de la vida de una persona va a requerir visitar diferentes especialistas de la salud, estos expertos pueden estar ubicados en diferentes instituciones y cada visita genera un nuevo registro en la historia clínica, que puede resultar muy valioso como consulta para el siguiente especialista que atienda a la persona. Para permitir que la historia clínica puede ser usada por las diferentes organizaciones de salud y especialistas se han establecido diferentes parámetros y estándares sobre información mínima a solicitar y formatos de captura que facilitan la coincidencia de los registros, aunque se tengan diferentes proveedores tecnológicos.

A medida que más fuentes de datos se generan se hace necesario contar con más técnicas que permitan encontrar entidades entre las diferentes bases de datos (Brunner & Stockinger, 2020).

Justificación

A pesar de que muchas técnicas han sido propuestas para resolver la búsqueda de entidades coincidentes, registros duplicados o integración de datos, aún hay espacio para proponer nuevos enfoques que consideren: el volumen de los registros a revisar, el tiempo requerido para procesar los registros, y en el caso del aprendizaje de máquinas: el tiempo de entrenamiento y conjunto de datos etiquetados para el entrenamiento y validación (Paganelli, Del Buono, Guerra, Pevarello, & Vincini, 2021). Algunas de estas técnicas pueden emplear métodos bayesianos aprovechando los avances en estadística y aprendizaje de máquina para abordar los problemas de escalabilidad y falta de información etiquetada.

Ante la creciente cantidad de bases de datos heterogéneas presentes disponibles para resolver un problema en un dominio dado, los diferentes estándares seguidos por los proveedores de las bases de datos para almacenar los registros y sistematización y procesamiento de la información, es conveniente realizar una revisión de técnicas disponibles para abordar el problema de detección de entidades coincidentes.

Objetivos

Objetivo general

Evaluar el uso de métodos bayesianos con aprendizaje automático que permita medir la coincidencia de entidades en bases de datos heterogéneas y en ausencia de un identificador de registro único y probar una metodología que emplee medidas de similitud de forma escalable para grandes conjuntos de datos.

Objetivos específicos

- Comparar el desempeño de los algoritmos de aprendizaje naive bayes y algunos del estado del arte para la clasificación de entidades.
- Emplear diferentes funciones de similitud de texto como características para el entrenamiento de los modelos de aprendizaje automático.
- Comparar el método de generación de datos sintéticos basado en cópulas con el método basado en redes bayesianas.

MARCO TEÓRICO Y ESTADO DEL ARTE

A pesar del esfuerzo investigativo, la vinculación de entidades en bases de datos heterogéneas sigue siendo un reto importante para abordar en los sistemas de información. Hay definiciones que datan de 1947 sobre la necesidad de conectar registros de las personas generadas en diferentes momentos de su vida, y por diferentes instituciones: como el acta de nacimiento y el acta de defunción (Marshall, 1947), esto resultaba difícil pues no se había planteado temas de estandarización para los formatos a nivel nacional ni se tenía un acuerdo de los campos requeridos de información sobre la persona.

Desde 1959 se tienen definiciones más específicas sobre la automatización de la detección de coincidencias empleando el computador (Newcombe, Kennedy, Axford, & James, 1959) con técnicas desde la coincidencia fonética en nombres y apellidos para abordar los problemas ortográficos en los registros, y combinación de los apellidos y nombres de la persona para tratar de encontrar el vínculo entre dos o más personas en diferentes bases de datos.

Un ejemplo de este problema se presenta con los siguientes registros:

Dadas las entidades:

$$x = (Nombre = Sebastian\ Lopez\ Valencia, \quad Telefono = 601523585, \\ Ubicacion = Envigado, Colombia)$$

Y

$$y = (Nombre = Sebastian\ L. Valencia, \\ Teléfono = +57\ 601\ 523\ 585, Ubicacion = Envigado)$$

Donde:

$$x \in B1 \wedge y \in B2 \wedge B1 \neq B2$$

Siendo

$$B1, B2: bases\ de\ datos$$

¿Cómo podemos encontrar que $x = y$?

Del ejemplo anterior, se puede definir que: Nombre, Teléfono y Ubicación son los atributos de la entidad y que se requieren conocer para determinar si dos entidades son iguales.

La gran cantidad de nuevas fuentes de datos, estándares de almacenamiento que aplican para diferentes países o regiones, la evolución en los sistemas de información y cambios en las definiciones de la organización son algunas de las causas de que aún se requieran investigaciones sobre nuevos métodos para integrar fuentes heterogéneas que permitan responder de forma más integral y completa sobre los casos de negocio o interrogantes. Un ejemplo práctico se da cuando dos compañías de un mismo sector se fusionan: ¿Cómo deben de conciliar su base datos de clientes? Es probable que compartieran clientes y que la información que hace referencia a un mismo cliente resulte contradictoria. (Laurenza, 2015).

A continuación, se detallan algunas técnicas recopiladas para abordar el problema de la coincidencia de registros o detección de entidades similares en bases de datos heterogéneas, y bajo ausencia de llaves de relacionamiento.

Coincidencia en base a reglas

Estas técnicas explícitas, y que pueden reflejar el conocimiento de un experto de dominio de la información, suelen ser reglas en base al contenido o texto que pueden aplicarse a las entidades. Una forma popular para decidir sobre la coincidencia de dos registros o entidades consiste en calcular un puntaje de similaridad en base a los atributos de las entidades.

Estas técnicas suelen asociarse con medidas de distancia de texto. Dado que se han propuesto muchas medidas para calcular la distancia entre palabras o texto, es conveniente definir función de similaridad para este trabajo:

Dada una función de distancia entre los textos x, y definido como: $d(x, y) = z$ donde $z \in \{\mathbb{R}^+\}$ y mientras z sea un valor mayor se dirá que los textos x, y son menos coincidentes. Se va a estandarizar la función de similaridad S como:

$$S(x, y) = 1 - \frac{d(x, y)}{max(longitud(x), longitud(y))}$$

Ecuación 1 Definición de similaridad

Donde, *longitud* cuenta la cantidad de caracteres de un texto.
Esta transformación será útil para ajustar cada función de distancia vista a continuación en una función de similaridad (Doan, Ives, & Halevy, 2012).

Funciones de distancia para texto

Dados dos textos a, b:

Tabla 1 Descripción de medidas de distancia

Nombre de medida	Descripción de medida	Tipo de medida	Ejemplo de aplicación de la medida
Edición o Levenshtein	Cantidad de pasos como: inserción, borrado y sustitución requeridos entre a y b para ser iguales.	Basado en secuencia	a = "David"; *b* = "Dave"; d(*a*,*b*)=2.0
Needleman-Wunch	Generalización de Levenshtein introduciendo la penalización por "ausencia" de caracteres u operaciones de inserción o borrado a un valor arbitrario.	Basado en secuencia	$ausencia = 0.5$ a = "David"; *b* = "Dave"; d(*a*,*b*)=2.0
Brecha afín (Affine Gap)	Una extensión de Needleman-Wunch que disminuye la penalización ante la ausencia de caracteres en secuencia.	Basado en secuencia	$ausencia = 1$ $ausencia\ prolongada = 0.5$ a = "David Smith"; *b* = "David Richardson Smith"; d(*a*,*b*)=5
Smith-Waterman	Introduce cambios en Needleman-Wunch para ignorar ciertos prefijos o sufijos ausentes en alguno de los textos a comparar.	Basado en secuencia	$ausencia = 1$ $ausencia\ prolongada = 0.5$ a = "Prof. David R. Smith"; *b* = "David Smith"; d(*a*,*b*)=8

Nombre de medida	Descripción de medida	Tipo de medida	Ejemplo de aplicación de la medida
Jaro	Comparación de textos cortos, basado en caracteres "similares" y cercanos, la penalización es menor ante la ausencia de algún carácter si se compara con Levenshtein.	Basado en secuencia	$a =$ "MARTHA"; *b* = "MARHTA"; d(*a,b*)=0.94
Jaro-Winkler	Diseñada para capturar casos en los que dos textos comparten un prefijo y, la medida introduce dos parámetros: PL, que es la longitud del prefijo común más largo entre los dos y PW, que es el peso para asignarle al prefijo	Basado en secuencia	$PW = 0.25$; $a =$ "$I\ am\ looking\ for\ a\ good\ book\ on\ machine\ earning$."; b = "I am looking for *b*="$I\ am\ looking\ for$ *a book on machine learning.*"; d(*a,b*)=0.97
Medidas de superposición (Overlap measure)	En base a dos textos, se generan los tokens para cada texto y luego cuentan la cantidad de tokens que se interceptan de ambos textos	Basado en conjuntos	$a =$ "$I\ am\ looking\ for\ a\ good\ book\ on\ machine\ learning$."; *b* = "*I am looking for a book on machine learning.*"; d(*a,b*)=0.89
Jaccard generalizado	Cuando los tokens (grupo de caracteres) no coincidan exactamente en todos los conjuntos, es posible especificar una función adicional que pueda asignar la similaridad entre los tokens y resolver este problema.	Basado en conjuntos	Usando una función de similaridad que permita omitir el error en uno de los caracteres: $a =$ "$I\ am\ looking\ for\ a\ good\ book\ on\ machine\ learning$."; b = "*I am looking for a god book or machine learning.*"; d(*a,b*)=0.8
TF / IDF	Basado en un corpus de texto amplio, si hay términos similares, pero no ampliamente repetidos va a determinar una similaridad mayor. Muy útil para análisis bajo contexto amplio o cuando hay entidades con mucha información.	Basado en conjuntos	Usando una función de similaridad que permita omitir el error en uno de los caracteres: $a =$ "$I\ am\ looking\ for\ a\ good\ book\ on\ machine\ learning$."; b = "*I am looking for a good book related to statistics.*"; d(*a,b*)=1
Soundex	Mapea los sonidos de las palabras a caracteres similares de forma que si la pronunciación coincide pueda encontrar similitud. Muy útil en el caso de nombres y apellidos cuando se detecta problemas de ortografía.	Basado en la pronunciación	$a =$ "$Anders$"; *b* = "$Anthers$"; d(*a,b*)=0

Nombre de medida	Descripción de medida	Tipo de medida	Ejemplo de aplicación de la medida
	Existen variaciones en lenguaje español como: PhoneticSpanish, Spanish Metaphone y MetaSoundex (Echeverri Calderón, 2022).		
Relación parcial (fuzzy wuzzy)	Busca en base al texto más corto que esté contenido una o más veces en el texto más largo.	Mixto (secuencial y conjuntos)	$a = "Apple\ Inc"$; $b = "Apple"$; d(*a*,*b*)=0
Monge-Elkan	Divide en sub-textos en base a la longitud del texto menor, luego usa una segunda función de similaridad para comparar los subtextos y en base a la coincidencia hallar si hay intercepción de subconjuntos.	Mixto (secuencial y conjuntos)	$a = ["Hello", "World"]$; $b = ["Hell0", "W0rld"]$; d(*a*,*b*)=0.83

Ahora, se va a realizar una explicación más detallada de algunas de las métricas:

Edición o Levenshtein: Para ilustrar cómo funciona la distancia de Levenshtein, consideremos los textos "David" y "Dave".

- Inicialización de la matriz: Se crea una matriz donde las filas representan los caracteres de la primera cadena ("David") y las columnas representan los caracteres de la segunda cadena ("Dave"). La matriz se inicializa de manera que la primera fila y columna contengan valores incrementales (0, 1, 2, ...).

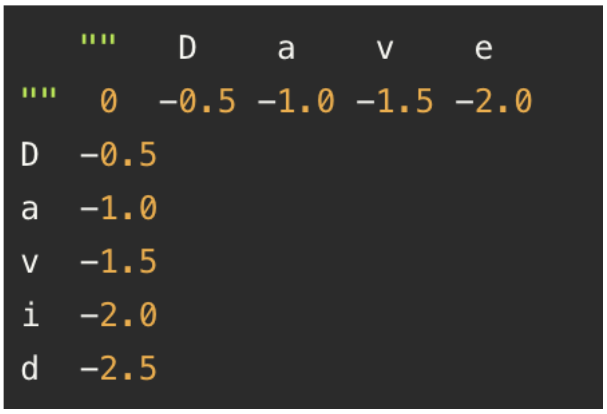
- Relleno de la matriz: Se procede a llenar la matriz calculando el costo mínimo de edición para cada par de caracteres. Se compara cada carácter de "David" con cada carácter de "Dave" y se aplica la siguiente fórmula:
  - Si los caracteres son iguales, el costo es el mismo que el de la celda diagonalmente anterior.
  - Si los caracteres son diferentes, el costo es 1 + el mínimo de los valores de las celdas adyacentes (arriba, izquierda y diagonal).

	""	D	a	v	e
""	0	1	2	3	4
D	1	0	1	2	3
a	2	1	0	1	2
v	3	2	1	0	1
i	4	3	2	1	1
d	5	4	3	2	2

- Resultado: La distancia de Levenshtein entre "David" y "Dave" se encuentra en la celda inferior derecha de la matriz, que es 2. Esto indica que se necesitan al menos 2 operaciones de edición para transformar "David" en "Dave". Las operaciones específicas en este caso podrían ser:
    - Eliminar 'i' de "David".
    - Eliminar 'd' de "Davi".

  Por lo tanto, la distancia de Levenshtein entre los textos "David" y "Dave" es 2.

Needleman-Wunch: Se crea una matriz donde las filas representan los caracteres de la primera cadena ("David") y las columnas representan los caracteres de la segunda cadena ("Dave"). La primera fila y columna se inicializan con valores incrementales multiplicados por la penalización de brecha (0.5).



	""	D	a	v	e
""	0	-0.5	-1.0	-1.5	-2.0
D	-0.5				
a	-1.0				
v	-1.5				
i	-2.0				
d	-2.5				

- Relleno de la matriz: La matriz se llena utilizando una puntuación de coincidencia (match), desajuste (mismatch), y penalización por espacio (penalización por brecha). Para cada celda (i, j), el valor se calcula utilizando la siguiente fórmula:
    - Si los caracteres coinciden, se suma la puntuación de coincidencia al valor de la celda diagonalmente anterior.
    - Si los caracteres no coinciden, se suma la puntuación de desajuste al valor de la celda diagonalmente anterior.
    - Además, se consideran las penalizaciones por espacios al sumar la penalización por brecha a los valores de las celdas adyacentes (arriba y a la izquierda).

	""	D	a	v	e
""	0	−0.5	−1.0	−1.5	−2.0
D	−0.5	1	0.5	0	−0.5
a	−1.0	0.5	2	1.5	1.0
v	−1.5	0	1.5	3	2.5
i	−2.0	−0.5	1.0	2.5	2.5
d	−2.5	−1.0	0.5	2.0	2.0

Después de llenar la matriz, se realiza el retroceso para obtener la mejor alineación global. Se comienza desde la celda inferior derecha y se sigue el camino con la puntuación más alta, moviéndose hacia arriba, a la izquierda o en diagonal, dependiendo de cuál tenga el valor más alto.

En este caso, el alineamiento final sería:

```
David
Dav-e
```

Por lo tanto, la distancia de Needleman-Wunsch entre los textos "David" y "Dave", con una penalización de 0.5 para inserciones y eliminaciones, indica que la mejor alineación global entre las dos secuencias tiene una puntuación de 2.0.

Affine Gap: Se utilizan tres matrices: M, X, y Y.

- M[i][j]: Representa la mejor puntuación de alineación hasta la posición (i, j) sin considerar una brecha.
- X[i][j]: Representa la mejor puntuación de alineación hasta la posición (i, j) considerando una brecha en la secuencia vertical.
- Y[i][j]: Representa la mejor puntuación de alineación hasta la posición (i, j) considerando una brecha en la secuencia horizontal.

Las matrices se rellenan utilizando las siguientes fórmulas para cada posición (i, j):

- (X[i][j] = max(M[i-1][j] - penalización de apertura, X[i-1][j] – penalización de continuación))
- (Y[i][j] = max(M[i][j-1] - penalización de apertura, Y[i][j-1] - penalización de continuación))
- (M[i][j] = max(M[i-1] [j-1] + (match/mismatch), X[i][j], Y[i][j]))

Suponga que se utiliza las siguientes puntuaciones:

- Coincidencia (match): +1
- Desajuste (mismatch): -1
- Penalización por apertura de brecha (penalización de apertura): -1
- Penalización por extensión de brecha (penalización de continuación): -0.5

La puntuación en la celda inferior derecha de la matriz M, X o Y refleja la calidad de la alineación óptima entre las dos secuencias. En este ejemplo, la puntuación

final se obtiene considerando las penalizaciones de apertura y extensión de brecha, resultando en una alineación óptima con la menor penalización total posible.

Smith-Waterman: Para los textos dados: a = "David Smith" y b = "Prof. David R. Smith", y con las penalizaciones especificadas (penalización = 1 y penalización prolongada = 0.5), se seguirán los siguientes pasos:

- Inicialización de la matriz de puntuación: Se crea una matriz (H) donde (H[i][j]) representa la puntuación del mejor alineamiento local de las subcadenas (a[1:i]) y (b[1:j]). La matriz se inicializa con ceros.
- Rellenado de la matriz: Se rellenan los valores de la matriz (H) utilizando las siguientes reglas:

$$H[i][j] = max\,(H[i-1][j-1] + match/mismatch\ score, H[i-1][j] - penalización\ por\ brecha\ H[i][j-1] - "penalización\ por\ brecha).$$

  Donde la puntuación de "match" es positiva (por ejemplo, +1) y la de no "coincidencia" es negativa (por ejemplo, -1).
- Aplicación de penalizaciones:
  - Penalización (1): Penalización por introducir un gap (espacio).
  - Penalización prolongada (0.5): Penalización adicional por extender una brecha.
- Cálculo de la matriz:
  - Para (i = 1) hasta (longitud de a):
    - Para (j = 1) hasta (longitud de b):
      Si (a[i-1] = b[j-1]), entonces (match/mismatch score = 1), de lo contrario, (match/mismatch score = -1).
  - Calcular (H[i][j]) usando las reglas descritas.
  - Búsqueda del máximo: El valor máximo en la matriz (H) representa la puntuación del mejor alineamiento local entre las subcadenas de (a) y (b).

El alineamiento óptimo para las secuencias "David Smith" y "Prof. David R. Smith" tendría en cuenta las coincidencias y las penalizaciones, mostrando una alta similitud especialmente en la parte "David Smith".

Jaro-Winkler: Se parte de la formulación de la distancia de texto de Jaro: $Jaro(s,t) = \frac{1}{3}\left(\frac{|m|}{|s|} + \frac{|m|}{|t|} + \frac{|m-t|}{|m|}\right)$ donde m es el número de caracteres coincidentes, t es el número de transposiciones (caracteres que coinciden, pero están en un orden diferente). Luego, Jaro-Winkler busca dar mayor peso a los prefijos comunes: $JaroWinkler(s,t) = Jaro(s,t) + \left(l \cdot p \cdot \left(1 - Jaro(s,t)\right)\right)$ donde l: Es la longitud del prefijo común al inicio de las cadenas, hasta un máximo de 4 caracteres; p es peso del prefijo.

Overlap o Superposición: Dados dos conjuntos de tokens A, B, se define como: $\text{Overlap}(\mathrm{A},\mathrm{B}) = \frac{|\mathrm{A}\cap\mathrm{B}|}{\min(|\mathrm{A}|,|\mathrm{B}|)}$. Pasos para calcular dados los textos:

$$\mathrm{A} = \mathrm{I},\ \mathrm{am},\ \mathrm{looking},\ \mathrm{a},\ \mathrm{good},\ \mathrm{book},\ \mathrm{on},\ \mathrm{machine},\ \mathrm{learning}.$$
$$\mathrm{B} = \mathrm{I},\ \mathrm{am},\ \mathrm{looking},\ \mathrm{for},\ \mathrm{a},\ \mathrm{book},\ \mathrm{on},\ \mathrm{machine},\ \mathrm{learning}.$$

Las palabras comunes entre los conjuntos (A) y (B) son:

$$A \cap B = \text{I, am, looking, a, book, on, machine, learning.}$$
$$|A \cap B| = 8$$

Las cardinalidades de A y B son:

$$|A| = 9$$
$$|B| = 9$$

El resultado es:

$$\text{Overlap}(A, B) = \frac{8}{\min(9,9)} = \frac{8}{9} \approx 0.89$$

Jaccard generalizado: Dados dos conjuntos de tokens A, B, se define la distancia de Jaccard como:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Los pasos seguidos son:

- Tokenizar: Dividir ambos textos en tokens (palabras individuales).
- Construcción de conjuntos: Crear conjuntos de palabras únicas para cada texto.
- Intersección y unión de conjuntos: Palabras que están en ambos conjuntos y Palabras que están en al menos uno de los conjuntos.

Siendo:

$$A = \textit{I am looking a good book on machine learning.}$$
$$B = "I\ am\ looking\ for\ a\ book\ on\ machine\ learning."$$

- Cardinalidad de la intersección: $|A \cap B| = 8$
- Cardinalidad de la unión: $|A \cup B| = 10$

$$J(A, B) = \frac{8}{10} = 0.8$$

TF/IDF: Siendo:

$$A = "\textit{I am looking a good book on machine learning.}"$$
$$B = "I\ am\ looking\ for\ a\ book\ on\ machine\ learning."$$

Se realizan los siguientes pasos:

- se debe construir un vocabulario con todas las palabras únicas presentes en ambos textos: Vocabulario = "I", "am", "looking", "a", "good", "book", "on", "machine", "learning", "for".
- Calcular la Frecuencia de Término (TF): se calcula como el número de veces que una palabra aparece en un documento dividido por el número total de palabras en ese documento

$$\text{TF}(\text{I}, a)\& = \frac{1}{9}, \&\text{TF}(\text{am}, a)\& = \frac{1}{9}, \&\text{TF}(\text{looking}, a)\& = \frac{1}{9}, \ \text{TF}(\text{a}, a)\& = \frac{1}{9}, \&\text{TF}(\text{good}, a)\& = \frac{1}{9}, \&\text{TF}(\text{book}, a)\& = \frac{1}{9}, \ \text{TF}(\text{on}, a)\& = \frac{1}{9}, \&\text{TF}(\text{machine}, a)\& = \frac{1}{9}, \&\text{TF}(\text{learning}, a)\& = \frac{1}{9}.$$

- Calcular la Frecuencia Inversa de Documentos (IDF): La frecuencia inversa de documentos (IDF) se calcula como el logaritmo del número total de documentos divididos por el número de documentos que contienen la palabra.

$$\text{IDF}(t) = \log\left(\frac{N}{|d \in D : t \in d|}\right)$$

Ejemplo:

$$\text{IDF(good)} = \text{IDF(for)} = \log\left(\frac{2}{1}\right) = \log(2) \approx 0.693$$

- Calcular el TF-IDF:
  El valor TF-IDF se obtiene multiplicando el valor TF por el valor IDF para cada palabra en cada texto.
- Representación de los Vectores TF-IDF: Ahora se representa cada texto como un vector en el espacio de términos y con una medida de distancia de vectores como la distancia de coseno se obtiene la distancia final.

$$VectorA = \mathbf{v_a} = [0,0,0,0,0.077,0,0,0,0,0]$$
$$VectorB = \boldsymbol{v_b} = [0,0,0,0,0,0,0,0,0,0.069]$$
$$\text{Distancia del coseno} = 1 - \frac{\mathbf{v_a} \cdot \mathbf{v_b}}{|\mathbf{v_a}||\mathbf{v_b}|}$$
$$\text{Distancia del coseno} = 1 - \frac{0}{0.077 \times 0.069} = 1 - 0 = 1$$

Aunque la distancia del coseno basada en TF-IDF resulta en una distancia de 1, es importante considerar las limitaciones de esta métrica. En este caso, la ausencia de coincidencias en las palabras con valores significativos de TF-IDF (debido a la IDF de "good" y "for") hace que los vectores resulten ortogonales en el espacio de términos, resultando en la máxima distancia posible.

Soundex: Para explicar la medida de distancia de texto utilizando Soundex con los textos (a = "Anders") y (b = "Anthers"), se deben seguir los siguientes pasos:

- Convertir las palabras a su código Soundex: Esto varía acorde al lenguaje del texto:
  - Anders: A536
  - Anthers: A536
- Comparar los códigos Soundex:
  - Código Soundex de "Anders": **A536**.
  - Código Soundex de "Anthers": **A536**.

Ambos textos, "Anders" y "Anthers", tienen el mismo código Soundex: A536. Esto significa que, según el algoritmo Soundex, las palabras se consideran fonéticamente equivalentes. En otras palabras, Soundex sugiere que "Anders" y "Anthers" suenan muy similares o casi idénticas en inglés, a pesar de las diferencias en su ortografía.

Monge-Elkan: Para explicar la medida de distancia de texto de Monge-Elkan con los textos (a = ["Hello", "World"]) y (b = ["Hell0", "W0rld"]), seguiremos los siguientes pasos:

- Definir la Función de Similitud: La medida de Monge-Elkan se basa en una función de similitud entre pares de cadenas. En este caso, utilizaremos la distancia de Levenshtein para medir la similitud entre dos cadenas.

Calcular la Similitud de Monge-Elkan: La fórmula para la medida de Monge-Elkan es:

$$\text{Monge-Elkan}(A, B) = \frac{1}{|A|} \sum_{a \in A} \max_{b \in B} \text{Similitud}\,(a, b)$$

Ejemplo:
Similitud(Hello,Hell0) ≈ 80, (por la sustitución de 'o' por '0').

Hay otras medidas que pueden ser definidas como variaciones de las ya mencionadas como la distancia de Hamming, distancia de bolsa y variaciones con generalización o suavizado de TF/ID o Jaccard generalizado.

Si se parte de una serie de funciones de similaridad junto con el conocimiento de un experto de área, se puede formular reglas para la coincidencia de registros como la siguiente:

Similitud entre x, y:

$$0.5\, S_{nombre}(x,y) + 0.3\, S_{edad}(x,y) + 0.2\, S_{direccion}(x,y) + 0.1\, S_{profesion}(x,y)$$

Ecuación 2 Ejemplo de reglas de coincidencia

Siendo: nombre, edad, dirección y profesión, características presentes en las entidades, y las constantes son valores provistos por los expertos del dominio.
Estas reglas son en particular útiles cuando la información de los registros se recupera de documentos físicos, y se debe lidiar con errores de reconocimiento de caracteres (OCR).

Coincidencia basada en aprendizaje

Desde el aprendizaje automático se ha abordado el problema tanto con técnicas de aprendizaje supervisado como con técnicas de aprendizaje no supervisado.

En aprendizaje supervisado se parte de tomar las características de la entidad, para el caso de que se empleen funciones de distancias estas serían cada uno de los valores resultados de aplicar dichas funciones, al agruparlas se les dará el nombre de tuplas de las características, dos entidades, representadas por sus tuplas, se van a comparan y se convierten en las entradas del modelo. Se incluye una etiqueta para saber si los dos registros coinciden, pasando a un ejercicio de clasificación binaria (Doan, Ives, & Halevy, 2012). Entre los algoritmos empleados en la literatura se destaca el uso de máquinas de soporte vectorial (SVM por sus siglas en inglés), árboles de decisión y sus variaciones, modelos basados en Naive Bayes, aprendizaje profundo con redes neuronales e incluso técnicas de procesamiento de lenguaje natural cuando las entidades se relacionan con descripciones de productos (Ramezani, Ilangovan, & Kum, 2021).

Dentro de los métodos de aprendizaje automático supervisado basado en Bayes encontramos la familia de métodos basados en Naive Bayes donde se asume que entre cada una de las características se presenta independencia condicional. Presentan características interesantes como convergencia rápida y requerimiento de baja cantidad de datos para entrenar. Matemáticamente se define como: Dado una variable independiente y, y un vector de características dependientes de y, definido como: $(x_1, x_2, x_3, \ldots, x_n)$

Ecuación 3 Probabilidad de Bayes

$$P(\, y \mid x_1, \ldots, x_n\,) = \frac{P(y)\prod_{i=1}^{n} P(\, x_i \mid y\,)}{P(x_1, \ldots, x_n)}$$

En base a los datos de entrada y sus distribuciones, es posible emplear alguno de los tipos de clasificadores de Naive Bayes:

Naive Bayes Gaussiano: Se asume que los valores continuos asociados con cada clase tienen una distribución normal. El concepto de valores continuos en el contexto de Naive Bayes Gaussiano se refiere a características o variables que pueden tomar cualquier valor dentro de un rango continuo, en contraposición a valores discretos que solo pueden tomar ciertos valores específicos. En el modelo de Naive Bayes Gaussiano, se asume que estos valores continuos siguen una distribución normal (o distribución gaussiana), lo que permite modelar la probabilidad de observar ciertos valores dados una clase específica. Para aclarar este concepto en el contexto de la fórmula, primero se asume que cada característica ($\mathrm{x_i}$) sigue una distribución normal. Para cada característica (x_i), dada la clase ($\mathrm{C_k}$), se tiene:

Ecuación 4 Probabilidad NB Gaussiano

$$P(x_i \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_{k,i}^2}} exp\left(-\frac{(x_i - \mu_{k,i})^2}{2\sigma_{k,i}^2}\right)$$

Donde:

- ($\mu_{k,i}$) es la media de la característica (x_i) para la clase (C_k)
- ($\sigma_{k,i}^2$) es la varianza de la característica (x_i) para la clase (C_k).

Naive Bayes Multinomial: Asume que los valores asociados con cada clase presentan una distribución normal. Este comportamiento se puede encontrar al modelar corpus de texto donde se usa la frecuencia de las palabras para hablar de características. En el caso del clasificador Naive Bayes Multinomial, se asume que las características siguen una distribución multinomial, que es adecuada para datos discretos, como el conteo de palabras en un documento. Aquí, ($x = (x_1, x_2, \dots, x_n)$) representa el conteo de cada palabra en el documento. Los pasos para entrenar el clasificador son:

- La probabilidad previa de la clase (C_k) se estima como la proporción de documentos que pertenecen a la clase (C_k):

$$P(C_k) = \frac{\text{Número de documentos en la clase } C_k}{\text{Número total de documentos}}$$

- Probabilidad de las Características $P(x_i \mid C_k)$: la probabilidad de observar la característica (x_i) dado la clase (C_k) se estima como la proporción del conteo de (x_i) en documentos de la clase (C_k):

$$P(x_i \mid C_k) = \frac{\text{Conteo total de } x_i \text{ en documentos de la clase } C_k + \alpha}{\text{Conteo total de todas las palabras en documentos de la clase } C_k + \alpha \cdot V}$$

  Donde:
  - α: es un parámetro de suavizado (Laplace o Lidstone).
  - V: es el número de palabras distintas en el vocabulario.
- Para clasificar un nuevo documento x, se calcula la probabilidad posterior para cada clase (C_k). Una vez que se han calculado las puntuaciones para todas las clases, se selecciona la clase con la mayor puntuación como la clase predicha para el nuevo documento:

Ecuación 5 Probabilidad NB Multinomial

$$\hat{C} = \arg\max_{C_k} \text{score}\,(C_k)$$

Naive Bayes Complemento: Parte de su versión multinomial e incluye mejoras para tratar con conjuntos de datos no balanceados (muchos valores hacen referencia a alguna o algunas clases y pocos al restante). Se considera un clasificador más estable que el multinomial al considerar los pesos de cada clase, se utiliza la información de las clases complementarias en lugar de la clase en sí para calcular las probabilidades condicionales. La idea es que, al hacer esto, se pueden corregir algunos sesgos y mejorar la precisión del modelo. Los pasos adicionales son:

- Para cada clase (C_k), se define su clase complementaria ($C_{\sim k}$) que agrupa todos los documentos que no pertenecen a (C_k).
- Se calcula el conteo total de cada palabra en los documentos que pertenecen a la clase complementaria ($C_{\sim k}$):

$$\text{Conteo total de } \mathrm{x_i} \text{ en documentos de la clase complementaria } \mathrm{C_{\sim k}} = \sum_{\mathrm{C_j \neq C_k}} \sum_{\mathrm{doc \in C_j}} \text{conteo de } \mathrm{x_i} \text{ en doc}$$

- Se calcula la probabilidad condicional de cada palabra (x_i) dado la clase complementaria ($C_{\sim k}$). Esto se hace utilizando un suavizado (como el suavizado de Laplace):

$$\mathrm{P(x_i \mid C_{\sim k})} = \frac{\text{Conteo total de } \mathrm{x_i} \text{ en documentos de la clase complementaria } \mathrm{C_{\sim k}} + \alpha}{\text{Conteo total de todas las palabras en documentos de la clase complementaria } \mathrm{C_{\sim k}} + \alpha \cdot \mathrm{V}}$$

- Dado que estamos utilizando la clase complementaria, las probabilidades condicionales se invierten para calcular las puntuaciones de las clases originales:

$$\text{puntuación}(\mathrm{C_k}) = \log \mathrm{P}\,(\mathrm{C_k}) - \sum_{\mathrm{i=1}}^{\mathrm{n}} \log P\,(\,x_i \mid C_{\sim k}\,)$$

Finalmente, se selecciona la clase con la mayor puntuación como la clase predicha para el nuevo documento:

Ecuación 6 Probabilidad NB Complemento

$$\hat{\mathrm{C}} = \arg\max_{\mathrm{C_k}} \text{score}\,(\mathrm{C_k})$$

Naive Bayes Bernoulli: Se asume, además de la independencia entre las características, que las características se comportan como una distribución de Bernoulli, esto es: variables binarias. Este modelo es particularmente útil en el procesamiento de textos y para tareas como la detección de spam. A continuación, se detallan los pasos del clasificador:

- La probabilidad previa de la clase (C_k) se estima como la proporción de documentos que pertenecen a la clase (C_k):

$$\mathrm{P(C_k)} = \frac{\text{Número de documentos en la clase } \mathrm{C_k}}{\text{Número total de documentos}}$$

- Para cada característica (x_i), se calcula la probabilidad de que esté presente (x_i = 1) o ausente (x_i = 0) dado la clase (C_k):

$$\mathrm{P(x_i = 1 \mid C_k)} = \frac{\text{Número de documentos en la clase } \mathrm{C_k} \text{ que contienen } \mathrm{x_i} + \alpha}{\text{Número total de documentos en la clase } \mathrm{C_k} + 2\alpha}$$

$$\mathrm{P(x_i = 0 \mid C_k)} = 1 - \mathrm{P(x_i = 1 \mid C_k)}$$

Donde (α) es un parámetro de suavizado (Laplace o Lidstone) para evitar probabilidades cero.

- Para clasificar un nuevo documento, se calcula la probabilidad posterior para cada clase (C_k):

Ecuación 7 Probabilidad NB Bernoulli

$$P(C_k \mid x) \propto P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$$

Naive Bayes Categórico: Asume que las variables son discretas y no continuas. Lo anterior permite definir el universo de valores posibles que toma cada variable. La fórmula del teorema de Bayes se ajusta para cada clase (C_k):

$$P(C_k|X) = \frac{P(C_k) \cdot \prod_{i=1}^{n} P(x_i|C_k)}{P(X)}$$

La clasificación se realiza seleccionando la clase (C_k) que maximiza la probabilidad posterior ($P(C_k|X)$):

Ecuación 8 Probabilidad NB Categórico

$$\hat{C} = \arg\max_{C_k} \left[P(C_k) \cdot \prod_{i=1}^{n} P(x_i|C_k) \right]$$

En la práctica, $P(X)$ es constante para todas las clases y no influye en la decisión final, por lo que se puede omitir.

- $P(C_k)$ se estima como la proporción de instancias en la clase (C_k) en el conjunto de entrenamiento.
- $P(x_i|C_k)$ se estima como la proporción de veces que la característica (x_i) toma un valor específico dado que la clase es (C_k)

Un posible problema con los estimadores de probabilidad es que algunos valores de (x_i) pueden no aparecer en el conjunto de entrenamiento para una clase particular, lo que resultaría en una probabilidad cero y, por ende, invalidaría el producto. Para evitar esto, se utiliza una técnica llamada *suavizado* de Laplace (o aditivo), que consiste en agregar un pequeño valor constante a todas las cuentas de frecuencia antes de normalizarlas.

Redes bayesianas: Desde el enfoque con métodos bayesianos se han usado las redes bayesianas, que parten del conocimiento experto y los valores observados. Estos modelos capturan información sobre eventos independientes y eventos condicionales: probabilidad de que se presente un evento B, dado que se presentó un evento A (Laurenza, 2015). Para entender este concepto, se definen los siguientes eventos a modo de ejemplo. Los datos observados para la construcción de la tabla se omiten:

Probabilidad evento A: Una empresa está localizada al sur del país	
Sí: 25%	No: 75%

Probabilidad evento B: Una empresa tiene más de 100 empleados	
Sí: 30%	No: 70%

La empresa es o no es líder en base a los eventos A y B:

A	Sí	Sí	No	No
B	Sí	No	Sí	No
Líder Sí	75%	33.3%	16.66%	0
Líder No	25%	66.6%	83.33%	1

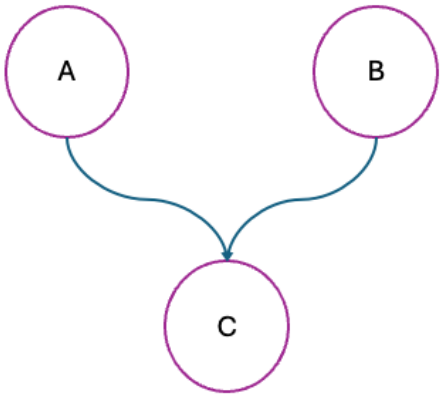


Ilustración 1 Grafo acíclico eventos A, B, C

Es común representar las redes bayesianas con gráficos acíclicos que permiten evidenciar las variables independientes y su efecto sobre las variables dependientes. También permiten limitar el grado de acción de unas variables sobre otras y reducir la cantidad de relaciones entre las variables. A diferencia de los modelos Naive, no se asume que las variables sean independientes, todo lo contrario: hay una estructura de interdependencia que debe abordarse por medio de las tablas de probabilidad condicional.

En base de que estos grafos y probabilidades constituyen un esfuerzo manual de generación y asunciones de un experto, algunos autores han propuesto algoritmos de formulación de estructuras de grafos y cálculo de probabilidades (Koller & Friedman, 2009) como PC (Constraint-Based Estimator en inglés) el cual es un algoritmo que ayuda a estimar la forma del grafo con las variables y sus probabilidades. Primero se encuentran las variables independientes, luego empieza a realizar combinaciones con las otras variables para llegar a un grafo de dependencias que cumpla con las restricciones calculadas usando los datos provistos. La dificultad de emplear las redes bayesianas radica en su costo computacional: Es un ejercicio en muchos casos clasificados como NP-Difícil: partiendo de la cantidad de variables, calcular las tablas de probabilidad puede convertirse en un problema de alto costo computacional de crecimiento polinómico.

El segundo problema radica en que, aun usando algoritmos como PC para llegar a la estructura del grafo, no hay garantía de alcanzarse la mejor solución posible, por lo que aún podría ser necesario la revisión de un experto sobre los grafos generados y valores obtenidos de probabilidad.

Aprendizaje no supervisado: Las técnicas más usadas son de agrupamiento (clustering en inglés) donde cada grupo va a contener las entidades que por similitud coinciden.

La idea básica con los algoritmos de agrupación es emplear una medida que pueda representar la afinidad entre dos entidades, las entidades con afinidad alta quedan localizadas en el mismo grupo. Este proceso termina cuando ya no es

posible unir más entidades o grupos de entidades porque la medida de afinidad no alcanza el umbral necesario para combinar las entidades.

En el caso de los algoritmos de agrupamiento, se puede encontrar ejemplo en la literatura del uso de DBSCAN, K-Means y AHCA (Akritidis, Fevgas, Bozanis, & Makris, 2020). Dentro de esta categoría está la propuesta de una resolución de entidades usando operaciones de agrupación previa de entidades que coinciden usando una variable auxiliar que apoya la tarea de particionar los registros en bloques, método de balanceo de bloques usando una estructura k-d de árbol, un muestreador de Gibbs distribuido parcialmente colapsado con algoritmos rápidos para realizar actualizaciones Gibbs (Marchant, Kaplan, Elazar, Rubinstein, & Steorts, 2021). La primera parte de estos enfoques hacen:

- Generación de bloques: emplea una asignación aleatoria primero y luego inferencia usando otras variables indicadas para relacionar las entidades.
- El árbol k-d o k-dimensional es una estructura que organiza los registros en un espacio euclídeo de k dimensiones, estas dimensiones consideran los valores de cada variable del registro como valores posibles, se puede tomar una variable y recorrer cada registro con su posible valor.

Procesamiento de lenguaje natural y Deep Learning: Otras propuestas consideran emplear campos como descripciones para obtener nuevas características y agrupar las entidades más similares por medio de algoritmos de clustering jerárquicos y características nuevas obtenidas por medio de análisis tipo NLP (Natural Language Procesing) como división por tokens y significado semántico con N-gramas (Akritidis, Fevgas, Bozanis, & Makris, 2020). En el aprendizaje profundo (Deep Learning en inglés) se usa esta propuesta, y el enfoque se basa en convertir las tuplas de características de las entidades a una representación de vectores numéricos que representan palabras o frases (Word embedding en inglés), esto permite preservar el significado semántico y pasar a una representación adecuada para un modelo matemático (Mudgal, y otros, 2018); luego, se pasa estas representaciones a la arquitectura de red neuronal que representa las características; pasa por las diferentes capas profundas y finalmente: la capa de clasificación, donde se obtiene si las dos tupas o entidades coinciden o no.

Dependiendo del autor, se han propuesto diferentes arquitecturas de redes neuronales en base a los resultados empíricos obtenidos, unas de las más usadas son las Redes Neuronales Recurrentes o RNN por sus siglas en inglés, las cuales han mostrado efectividad en tareas como reconocimiento de voz, traducciones y otros casos de procesamiento natural de lenguaje (NLP por sus siglas en inglés) (Mudgal, y otros, 2018).

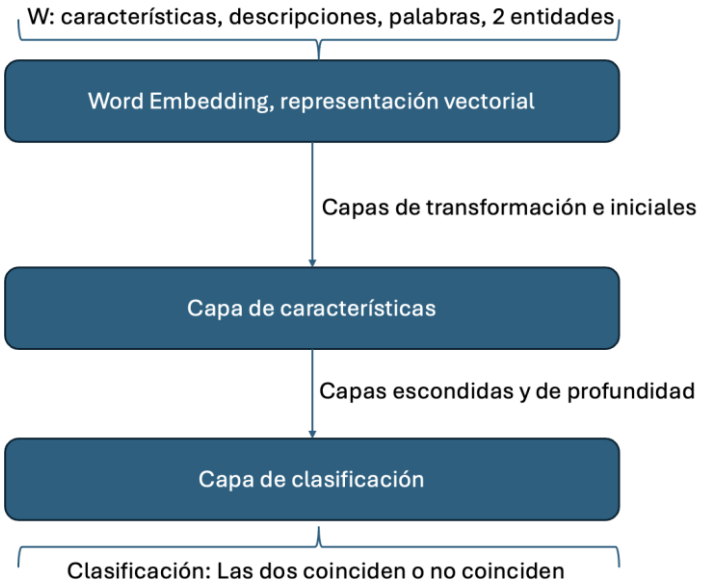


Ilustración 2 Red neuronal propuesta con embeddings de texto

Las RNN utilizan datos secuenciales (una frase o descripción puede ser representada por medio de secuencias, lo que permite preservar significados semánticos y sintácticos) y por su arquitectura, cuenta con memoria para complementar los datos de entradas previas, esto se explica porque en cada capa se comparten parámetros con sus capas anteriores (IBM, 2024).

Para convertir las características textuales como descripciones de productos o prosa libre, en valores numéricos que puedan usar los algoritmos se pueden usar algoritmos pre-entrenados en corpus públicos, como es el caso de GloVe (Global Vectors for Word Representation) y Word2vec, los cuales pueden encontrarse como modelos listos o pueden sen entrenados en un corpus diferente (Mudgal, y otros, 2018). La decisión de escoger entrenar nuevamente puede deberse a que se presenten muchas palabras que no estaban en el corpus del modelo base, esto puede ser un reto especialmente para idiomas diferentes del inglés, ya que será necesario entrenar sobre un corpus lo suficientemente representativo para obtener la representación vectorial de cada palabra.

Algunos autores han propuesto arquitecturas basadas en RNN y capas adicionales para tener memoria de corto-largo plazo (LSTM por sus siglas en inglés), que le permite aprender dependencias entre las secuencias de los datos de entrada por medio del entrenamiento (Ebraheem, Thirumuruganathan, Joty, Ouzzani, & Tang, 2018). Esta estrategia permite reducir los errores causados por la propagación de errores entre las capas, el cual es un problema muy conocido en estas arquitecturas por compartir parámetros en todas sus capas (ampliamente documentado en inglés como gradient vanishing problem).

Estrategias de optimización:

El costo computacional para resolver la coincidencia de entidades es alto en términos generales, lo que hace necesario el uso de técnicas de optimización que van desde emplear heurísticas, uso de medidas de distancia menos complejas, uso de algoritmos de aprendizaje de rápida convergencia y cambios de la aproximación para resolver el problema (Marchant, Kaplan, Elazar,

Rubinstein, & Steorts, 2021). A continuación, se describen algunas de estas técnicas.

Agrupar y luego comparar: Una forma de reducir la complejidad de comparación entre entidades, consiste en agrupar primero las entidades que podrían ser similares bajo algún criterio que emplee técnicas de bajo costo computacional, puede ser una similitud por prefijos o sufijos; una vez se tengan los diferentes grupos de entidades se procede a realizar la búsqueda de identidades en los subgrupos, esto reduce el espacio de búsqueda y comparación de un registro contra todo. Evidentemente, puede haber un riesgo de que ciertas entidades coincidentes queden en grupos distintos por lo que conviene el uso de técnicas generales que permitan agrupar más que discriminar. Este enfoque podría verse beneficiado de los marcos de trabajo para computación distribuida, lo cual permitiría tratar cada grupo en computadoras diferentes para lograr paralelismo en el trabajo.

Sistemas disponibles para coincidencia de entidades

Hay varios sistemas disponibles que realizan todo el ciclo: visualizar los datos, aplicar algunas transformaciones, corregir y mejorar precisiones, y procesar masivamente los documentos requeridos, **Magellan** es una herramienta para coincidencia de identidades desarrollada por la Universidad de Wisconsin y que se caracteriza por ser de código abierto, contiene lo requerido para aplicar reglas de calidad especificadas y generadas por medio de aprendizaje de máquina, se usa ampliamente a la hora de comparar con nuevas propuestas en términos de algoritmos o métodos, dado que se logra alta efectividad en los conjuntos de datos (Konda, y otros, 2018). Otras empresas como **SAS** e **Informática** ofrecen sus propias herramientas, normalmente de código cerrado.

DESCRIPCIÓN DE LOS DATOS

Adquisición de los datos

Aprovechando el auge de los datos abiertos que plataformas como **Kaggle** tienen a disposición, se emplea un conjunto pre-clasificado de productos de dos plataformas en línea de retail cuyo nombre aparece como: *Product Clustering, Matching & Classification* (Akritidis, Fevgas, Bozanis, & Makris, 2020). La industria del comercio electrónico ha experimentado un crecimiento continuo, lo que ha vuelto especialmente relevante el problema de encontrar el mismo producto entre las diferentes plataformas de e-commerce, para encontrar el mejor precio. A medida que más empresas trasladan sus actividades a la web, el volumen y la diversidad de la información relacionada con productos aumentan rápidamente. Cada conjunto de datos viene acompañado de su correspondiente archivo de referencia que se puede utilizar para evaluar su desempeño. El repositorio incluye 18 conjuntos de datos del mundo real de diferentes categorías de productos, adquiridos de dos plataformas de comparación de productos en línea: **PriceRunner** y **Skroutz**. Hay 8 conjuntos de datos para cada tienda, es decir: 16 conjuntos de datos representan una categoría específica de productos y 2 que representan la agregación para cada tienda.

Los conjuntos de datos se proporcionan en formatos estándar CSV y XML. Cada entrada en el CSV/XML incluye la siguiente información:

Tabla 2 Campos de la fuente de datos de retail

Identificador Campo	Descripción	Ejemplo de celda
ProductID	Identificador textual del producto con un resumen de características, es el campo de donde se extraen las características	amd ryzen 5 1600 box pliromi ke se eos 36 dosis
VendorID	Identificador numérico para encontrar al proveedor específico: Skrountz y PriceRunner son plataformas donde operan diferentes vendedores	1
ClusterID	Identificador numérico que asocia al producto con su producto global, es la etiqueta y forma de saber si varios productos corresponden al mismo.	14
ClusterLabel	Identificador textual que asocia al producto con su producto global, es la etiqueta y forma de saber si varios productos corresponden al mismo al leer el conjunto de datos.	Apple iPhone 8 Plus 64GB
CategoryID	Identificador numérico que permite relacionar el producto en una categoría de e-commerce como: Celulares, Bicicletas todo terreno, cámaras digitales, etc.	30
CategoryLabel	Identificador textual que permite relacionar el producto en una categoría de e-commerce como: Celulares, Bicicletas todo terreno, cámaras digitales, etc.	Fridges

Las categorías son: CPU, Cámaras Digitales, máquinas lavavajillas, hornos microondas, celulares, refrigeradores, televisores y lavadoras. Son 13233 productos y 306 vendedores.

Preprocesamiento de los datos

Tipo de máquinas empleadas:

- MacBook M3 Pro, 18GB de memoria y 512 GB de SSD. OS: Sonoma 14.5. Python 3.10. La cual se usó en la mayoría de los experimentos.
- Clúster de Databricks: Azure Databricks tipo: X-small. Driver Standard_E8ds_v4, workers Standard_E8ds_v4 x 2. Procesador: vCPU 8, Intel® Xeon® Platinum 8370C (Ice Lake) o Intel® Xeon® Platinum 8272CL. Memoria: 64GB. La cual se usó principalmente en la generación de los datasets con producto cartesiano.
- Asus AMD Ryzen 9 5900HS con GeForce RTX 3070 40GB RAM, 1TB SSD. Usada principalmente para generar los embeddings como características y entrenamiento de algunas redes neuronales con Pytorch.

Se opta por emplear los datos originales en CSV. El primer archivo revisado es el agregado de Price Runner con un total de 35311 artículos, luego el de Skroutz con un total de 238710 artículos. Al unir ambos conjuntos en uno solo, se encuentra una población de 74782 tipos de producto, algunos productos con solo una coincidencia y otros con hasta 100 coincidencias.

ClusterLabel	ProductID
Mitsubishi MSZ/MUZ-HJ25VA	100
Inventor Omnia Eco O3MVI32-09WiFiR / O3MVO32-09	98
Gree Bora GRS-121 EI/JBR1-N3	96
Inventor Omnia Eco O3MVI32-12WiFiR / O3MVO32-12	95
Mitsubishi MSZ/MUZ-HJ35VA	95
Gree Bora GRS-181 EI/JBR1-N3	95
Gree Bora GRS-101 EI/JBR1-N3	94
Inventor Omnia Eco O3MVI32-18WiFiR / O3MVO32-18	90
Gree Bora GRS-241 EI/JBR1-N3	90
Gree Lomo GRS 101 EI/JLM1-N3	88
Midea Prime 2018 MA2-09NXD0	87
Inventor Omnia Eco O3MVI32-24WiFiR / O3MVO32-24	87
Gree Lomo GRS 121 EI/JLM1-N3	87
Midea Prime 2018 MA2-12NXD0	86
Gree Lomo GRS 241 EI/JLM1-N3	85
Mitsubishi MSZ/MUZ-HJ50VA	85
Mitsubishi MSZ/MUZ-HJ60VA	83

Ilustración 3 Conteo de productos por sus agrupaciones

Al unir ambos conjuntos de datos, el CSV resultante alcanza un peso estimado de 28MB, al cambiar a formato parquet, alcanza 7MB, se opta este segundo formato para reducir el tamaño en disco. Todos los conjuntos de datos procesados se almacenan en parquet que permite una compresión mayor que CSV, y gracias a la amplia adopción de la comunidad, no presupone un riesgo para encontrar herramientas y bibliotecas que pueda leer este tipo de archivos.

Se realiza un intento de hacer un producto cruzado entre cada artículo, la idea es tener dos artículos por fila y calcular la columna de coincidencia o no coincidencia en base a la columna: ClusterID, que de coincidir indicaría que es

el mismo producto. No fue posible en MacBook luego de desbordarse por falta de memoria en 30 minutos de procesamiento del producto cartesiano. Se procede a intentar el producto cartesiano en Databricks, luego de 30 horas de procesamiento, se genera un desborde de memoria que imposibilita continuar, no se tuvo en cuenta que un conjunto así en producto cartesiano generaría más de 20 billones de registros.

Se procede a realizar la transformación de producto cartesiano en los subconjuntos de CPU, Cámaras Digitales y televisores para ambos retailers. El producto cartesiano sobre el conjunto original permite obtener un conjunto de datos formado por dos tuplas de entidades, que podrían o no coincidir, esto se hace dado que el conjunto de datos originales sólo presenta un producto por fila y se busca que cada fila tenga dos productos y su columna de coincidencia o no coincidencia.

A continuación, se resumen la cantidad de registros por conjunto de datos y coincidencia o no coincidencia de producto:

Tabla 3 Conteo de registros por subconjunto de datos

Conjunto	**Coinciden**	**No Coincide**
Skroutz_Camaras	75.525	16.824.796
Price_Camaras	19.169	7.254.640
Skroutz_tvs	142.702	48.885.302
Price_tvs	22.738	12.679.358
Price_cpus	11.840	14.903.204
Skroutz_cpus	31.762	3.601.074

Desde el inicio, se evidencia un alto desbalanceo entre las dos clases, una opción para abordar este problema es el uso de datos sintéticos o el uso de métodos de aprendizaje automáticos que puedan manejar el desbalanceo como el caso de Naive Bayes Complemento.

Se generan variaciones de los conjuntos de datos con las características basadas en medidas de distancia de texto y con embeddings como word2vec; este archivo alcanza el peso de 79.8MB debido a los valores únicos y poco repetidos de los vectores representativos de hasta 600 posiciones de valores decimales generados en el computador ASUS para aprovechar su GPU y en un tiempo de 55 minutos en total.

METODOLOGÍA DE TRABAJO

Se emplea la metodología Team Data Science Process (TDSP) de Microsoft (Corporation, 2024) la cual parte en su base del método iterativo para la minería de datos: CRISP DM, que fue originalmente propuesto por un consorcio de compañías de tecnología para la Unión Europea en 1996 (Cross Industry Standard Process for Data Mining, 2021) (Shearer, 2000). TDSP cuenta con una definición concreta de roles, herramientas, tareas durante el proyecto, entregables y cuenta con amplia documentación para la construcción de repositorios y ambientes de trabajo.

TDSP es un modelo iterativo pensado para realizar proyectos de Machine Learning y analítica avanzada, presupone la existencia de varios roles empresariales de los que se va a prescindir. Las razones para escoger esta metodología base son:

- Familiaridad y experiencia previa con esta metodología, lo que incluye tener plantillas, herramientas de gestión y práctica real en proyectos.
- La metodología cuenta con más refinamiento y detalle específico de sus etapas, entradas, salidas y roles. Todos estos aspectos están mejor detallados.
- Unas etapas más iterativas respecto a los ajustes en el entendimiento o definición del objetivo de la investigación, esto es importante teniendo en cuenta que la disponibilidad de la información podría requerir hacer ajustes sobre las preguntas detalladas de investigación.
- La posibilidad de entregar valor de forma incremental. TDSP toma en consideración varios principios ágiles y permite que se entregue valor de forma incremental. Para la investigación servirá para llevar a cabo validaciones sobre las decisiones tomada.

Se realizaron algunos ajustes a las etapas propuestas para la metodología de trabajo de forma que se adecúe mejor con la investigación y donde la validación de negocio o experto se realiza con apoyo la directora de investigación.

Etapas de la metodología:

Ilustración 4 Marco de trabajo TDSP en español

Imagen ajustada de (Corporation, 2024)

Entendimiento Empresarial

Validación de las preguntas de investigación y objetivos específicos. Se realizan las siguientes tareas:

- Se identifican bases de datos que permitan abordar algún escenario de registros o entidades duplicadas o de cruces entre fuentes heterogéneas. A continuación, se detallan los conjuntos de datos clasificados:

Tabla 4 Fuentes de datos consideradas

Base de datos	Descripción	Fuente	¿Está clasificado?
Product Clustering, Matching & Classification	Conjunto escogido para esta investigación	Kaggle	Sí
Disney, Hulu, Netflix, Prime movies & Show	Hay información de los títulos, casting, director, duración, país de origen, rating y otros que podrían usarse para hallar coincidencia entre las plataformas y encontrar las películas comunes	Kaggle	No
Benchmark datasets for entity resolution	Conjuntos de datos variados para clasificación de entidades, por su usabilidad se descartó.	Enlace	Algunos sí
Hugging Face: Entity Matching	Conjuntos de datos variados para clasificación de entidades, ricos en texto	HuggingFace	Sí

	y pre-etiquetados. Ya incluye la comparación entre productos.		
Fuzzy Name Matching	Conjuntos de datos con dos columnas: Nombre A y Nombre B, la idea es calcular la distancia de los nombres. No se indica si se trata de la misma entidad.	Fuzzy Name Matching (kaggle.com)	No
Amazon Fine Food Reviews	Este conjunto de datos consta de reseñas de alimentos de Amazon. Los datos abarcan un período de más de 10 años, incluidas las ~500.000 reseñas hasta octubre de 2012	Amazon Fine Food Reviews (kaggle.com)	No

Se evaluaron otras fuentes como datos.gov.co pero no fue posible dar con un conjunto o conjuntos que fueran fácilmente relacionables o estuvieras clasificadas las entidades para hallar coincidencias.

- Se identifican las variables necesarias para recopilar la información y su definición. Esto es, la exploración de los datos y la definición de las métricas de distancia de texto que se usarán como características.
    - ProductID y ProductID2 son las columnas usadas para comparar y extraer las distancias de texto propuestas:
        - Levenshtein.
        - Needleman Wunsch.
        - Affine Gap.
        - Smith Waterman.
        - Jaro.
        - Jaro Winkler.
        - Overlap Coefficient.
        - Generalized Jaccard.
        - TF/IDF.
        - Soft TF/IDF.
        - Fuzzy Wuzzy Partial Ratio.
        - Bag Distance.

      Las características escogidas permiten un cálculo en paralelo, se escogen en base a sus ventajas mencionadas en el Marco Teórico. Otro factor importante para considerar es la correlación entre variables. Para una mayor eficiencia en el modelo, se debería tomar un par de variables altamente correlacionadas y eliminar una de las dos.
    - ClusterID se usa para relacionar si ambos productos hacen referencia al mismo.

        - Features: Campo producto de la transformación vectorial de ProductID y ProductID2 usando Word2Vec y útil para probar redes neuronales según la literatura (Barlaug & Gulla, 2021).
    - Se identifican los orígenes de datos con los que ya se cuenta acceso y se enumera las fuentes faltantes.
    - **Salidas obtenidas**: Diccionario de datos, definición de objetivos y modelo a entrenar, descripción y acceso a los orígenes de datos.
    - **Adiciones metodológicas**: Acompañamiento por parte de directora sobre cuestiones temáticas: métodos estadísticos mencionados en las técnicas propuestas. Dentro de las sesiones se abordaron refuerzos sobre la teoría de Bayes, generación de datos sintéticos con Bayes, tablas condicionales y el valor de profundizar en técnicas para hallar la coincidencia de entidades en bases de datos heterogéneas.

Adquisición y comprensión de los datos

Resolución de problemas típicos de calidad de datos, montaje de ambiente de laboratorio y trabajo de cómputo, proposición de arquitectura para la solución de canalización de datos y repositorio para modelos y código.

- **Salidas esperadas**: Informe de calidad de datos, resolución de problemas detectados de calidad y desbalanceo, diagrama de arquitectura de la solución para implementación: desde la fuente, movimiento de datos, almacenamiento, procesamiento, consumo y despliegue de los modelos, evaluación de las fuentes de datos versus los objetivos de la investigación para revisar que se tenga lo necesario, primeros conjuntos de datos preparados y almacenados.

- **Adiciones metodológicas**: Uso de la interpretación de las medidas de similitud entre entidades, uso de las metodologías que se encuentran en el marco teórico sobre la captura de datos y volúmenes significativos de información. También se analiza la existencia de plataformas y bibliotecas que puedan usarse para resolver problemas de coincidencia de entidades (véase el marco teórico).

Modelado

Se validan las variables y se realiza la ingeniería de características para reducir, adicionar, modificar las variables del modelo, principalmente: funciones de similitud. Luego, se pone en marcha las actividades de entrenamiento para generar las primeras versiones de los modelos de Machine Learning.
Dado que se presenta un desbalanceo entre las dos clases en la tarea de clasificación: Coinciden, No Coinciden (véase en la descripción de los datos), se decide explorar la generación de datos usando dos alternativas multivariable:

Cópulas: Se emplea un método de generación de datos sintéticos no-paramétricos basados en cópulas: son funciones de distribución multivariable que explican la dependencia estructural sobre los datos (Restrepo, Rivera,

Laniado, Osorio, & Becerra, 2023), permite trabajar con distribuciones sin asumir que son paramétricas. Dado su uso de tablas de frecuencia y funciones de distribución de cada variable, resultó más rápido en la generación de más de un millón de registros sintéticos en una hora con el tipo de cómputo empleado (MacBook); a continuación, se muestra algunas gráficas de las características con los datos reales a la izquierda y a la derecha los datos generados con cópulas:

Levenshtein:

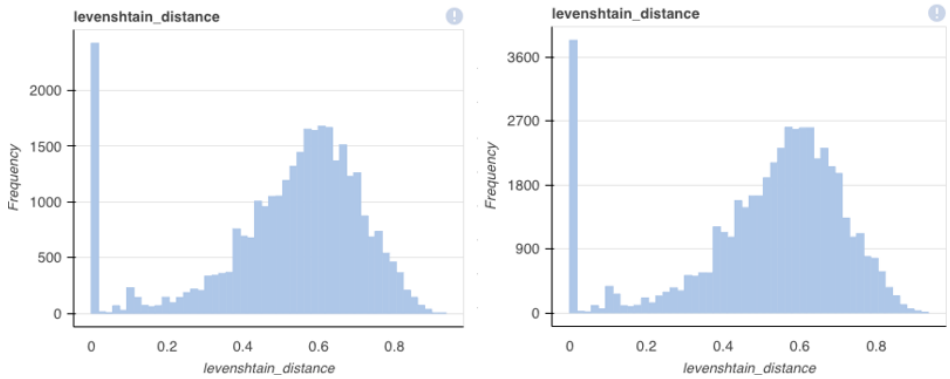


Ilustración 5 Comparación de las distribuciones para la distancia de Levenshtein entre los datos reales y los sintéticos

Tabla 5 Estadísticos de las distribuciones para Levenshtein

Estadístico	Datos reales	Datos sintéticos
Mínimo	0	1.0044×10^{-06}
Máximo	0.9362	0.9356
Q1	0.4386	0.4371
Q2 (Mediana)	0.5647	0.5637
Q3	0.6571	0.6572
Promedio	0.5174	0.5172
Moda	0.8000	No se determina

Jaro-Winkler:

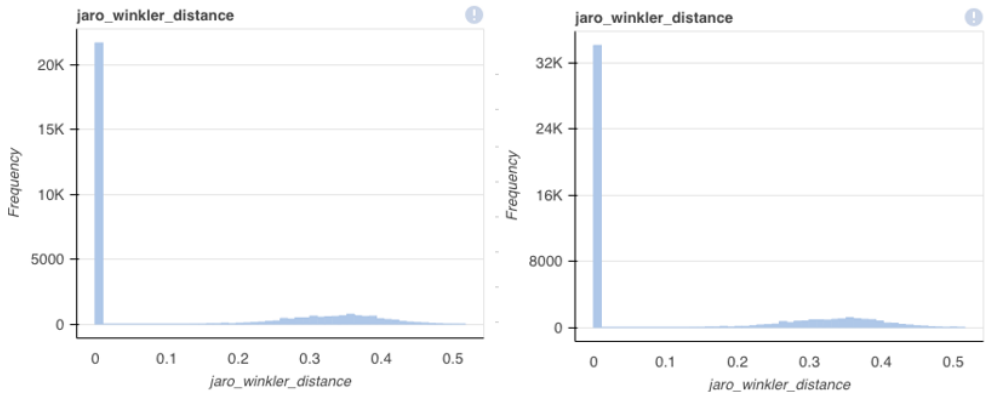


Ilustración 6 Comparación de las distribuciones para la distancia de Jaro-Winkler entre los datos reales y los sintéticos

Tabla 6 Estadísticos de las distribuciones para Jaro-Winkler

Estadístico	Datos reales	Datos sintéticos
Mínimo	0	3.6999×10^{-09}

Máximo	0.5174	0.5162
Q1	0	0.0019
Q2 (Mediana)	0	0.003801
Q3	0.2862	0.2861
Promedio	0.1070	0.1088
Moda	0	No se determina

Overlap:

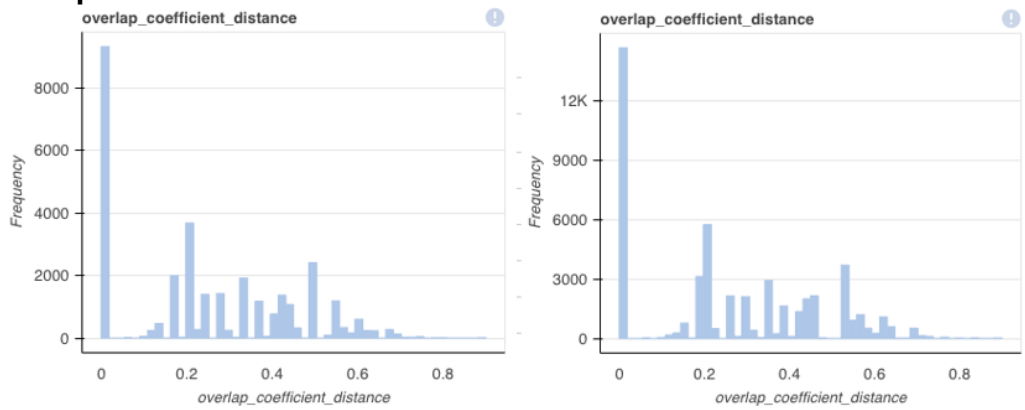


Ilustración 7 Distribuciones para la distancia del coeficiente de sobrelapamiento entre los datos reales y los sintéticos

Tabla 7 Estadísticos de las distribuciones para Solapamiento

Estadístico	Datos reales	Datos sintéticos
Mínimo	0	3.3556×10^{-07}
Máximo	0.900	0.8996
Q1	0	0.0076
Q2 (Mediana)	0.2222	0.2160
Q3	0.4286	0.4363
Promedio	0.2511	0.2631
Moda	1	No se determina

Bag:

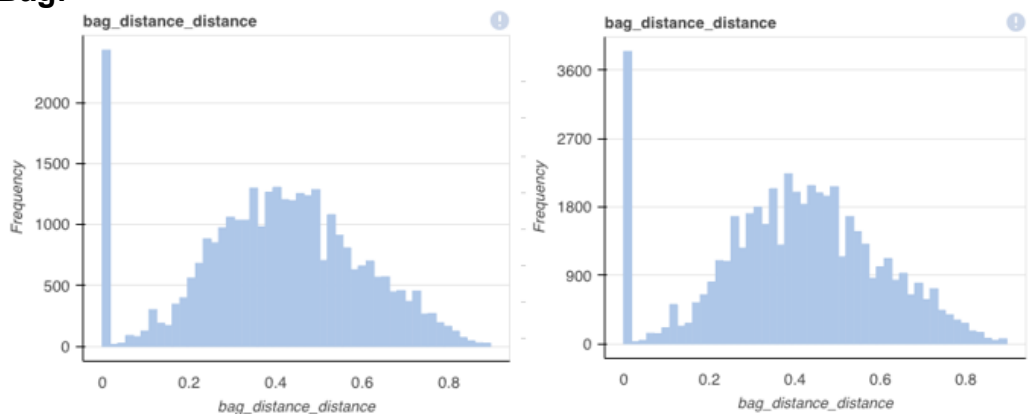


Ilustración 8 Distribuciones para la distancia del coeficiente de la distancia de bolsa (bag) entre los datos reales y los sintéticos

Tabla 8 Estadísticos de las distribuciones para Bag

Estadístico	Datos reales	Datos sintéticos
Mínimo	0	3.0794×10^{-06}

Máximo	0.8957	0.8956
Q1	0.2857	0.2867
Q2 (Mediana)	0.4103	0.4108
Q3	0.5333	0.5339
Promedio	0.4025	0.4033
Moda	0.5403	No se determina

Correlación de variables usando Kendall's-Tau como coeficiente no paramétrico: Los gráficos de correlaciones ayudan a entender la relación entre dos variables, por ejemplo: determinar si el aumento de peso se relaciona con el aumento de riesgo de ataques cardiacos, aunque dos variables estén altamente correlacionadas, esto no significa que haya relación de causalidad entre ellas. Se emplea la matriz de correlaciones para verificar que entre los datos reales observados, y los datos sintéticos generados, se mantengan las relaciones entre las variables. Se espera que algunas características mantengan una fuerte correlación dado que algunas son variaciones una de otra, como en el caso de Affine Gap y Needleman-Wunch.

Datos reales:

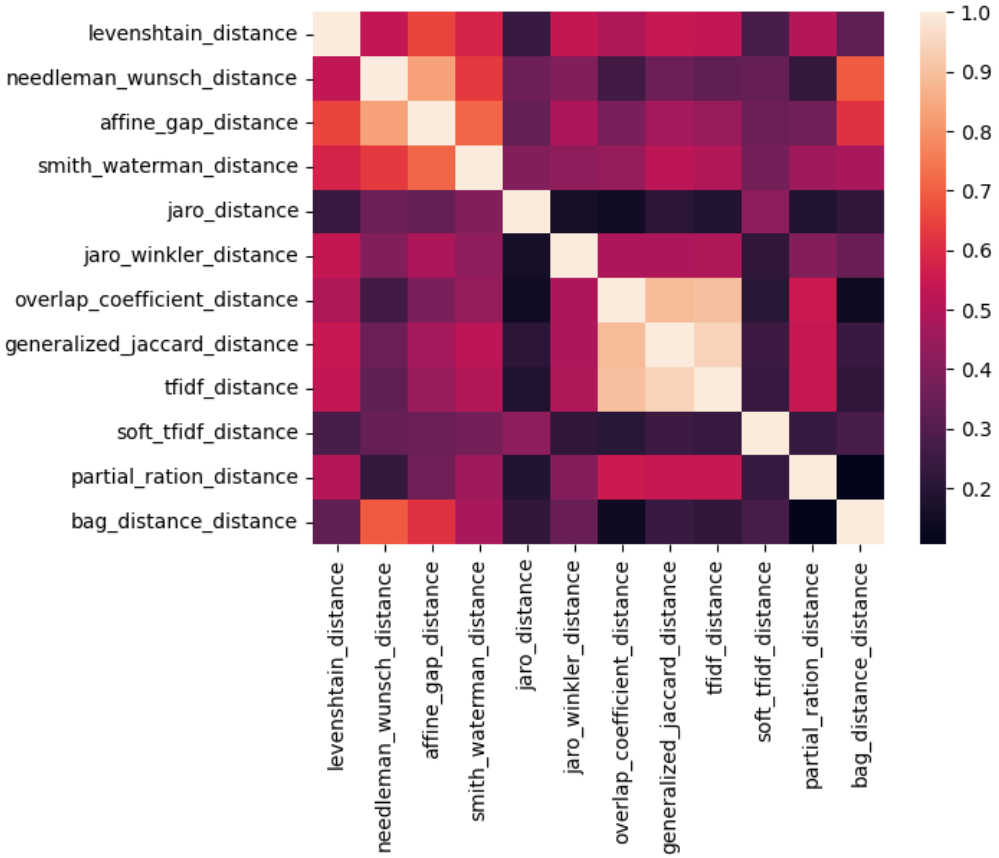


Ilustración 9 Matriz de correlación de los datos reales

Lo primero que se resalta sobre las diferentes medidas, es que algunas tienen una alta correlación positiva (valor cercano a 1.0 y valores más

cercanos a rojo oscuro), lo cual tiene sentido dado que muchas medidas de texto se relacionan. Needleman-Wunch se correlaciona con Levenshtein, también Affine Gap y Needleman-Wunch. Para un modelo de Machine Learning puede resultar mejor que las tuplas de características no se correlacionen, lo que significa eliminar una de las dos características.

Datos sintéticos generados por cópulas:

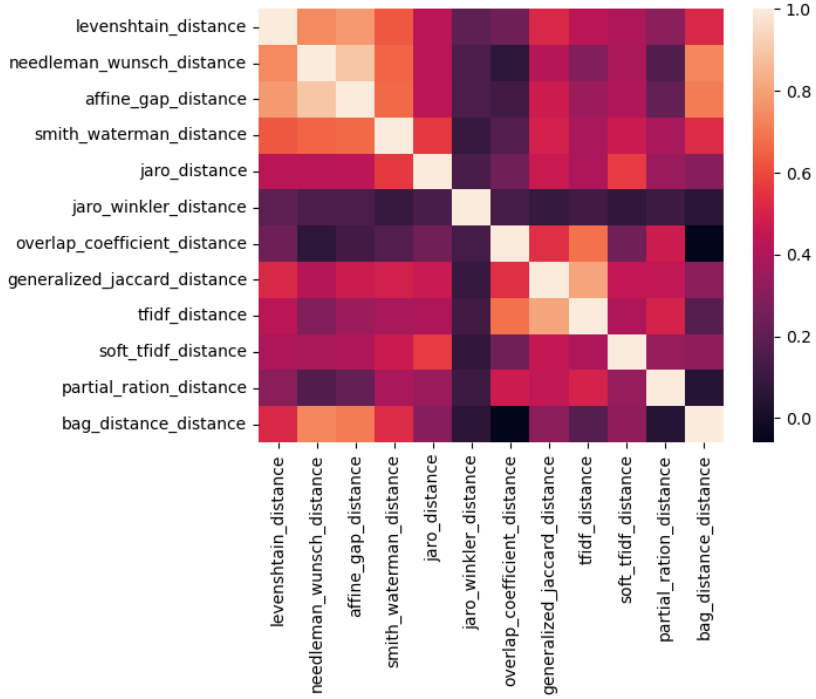


Ilustración 10 Matriz de correlación de los datos sintéticos por cópulas

Datos sintéticos generados por la red bayesiana:

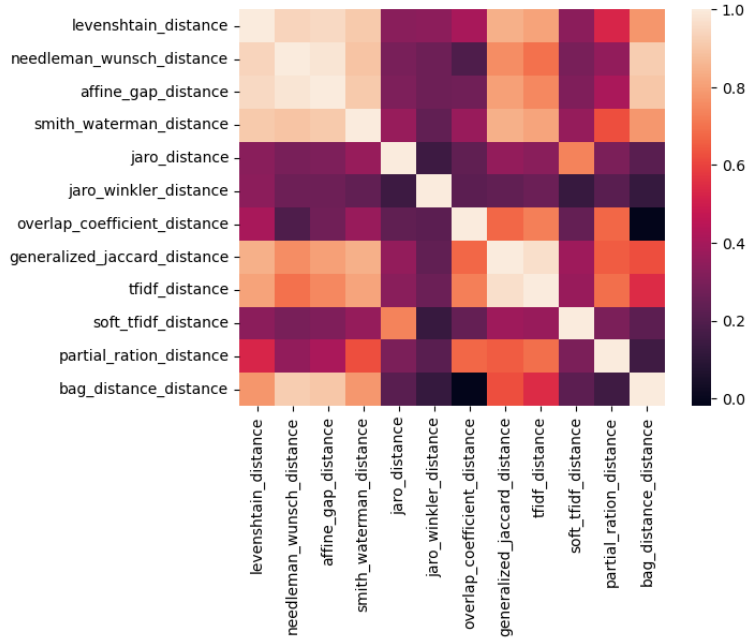


Ilustración 11Matriz de correlación de los datos sintéticos por red bayesiana

Se observan correlaciones similares en los datos sintéticos a los reales. Al revisar las matrices de correlaciones, se evidencia pequeñas variaciones en la relación entre las variables como Affine Gap y distancia Bag. Una tarea posterior podría incluir tomar los pares de variables más correlacionadas y continuar los experimentos con solo una variable de las dos: esto es una estrategia común en el aprendizaje de máquina.

Redes Bayesianas: Usando la biblioteca DataSynthetizer es posible emplear redes bayesianas que implementan la búsqueda inteligente dentro del espacio de soluciones o posibles combinaciones de las características como nodos dependientes o independientes (Ping, Stoyanovich, & Howe, 2017), lo que permite llegar a una red que simula la distribución multivariable de la solución, respetando las estructuras internas de los datos. Para acotar el dominio, es necesario limitar la cantidad de valores categóricos de las variables discretas y establecer un número máximo de nodos padres que puede tener una variable dada. Para el caso del conjunto de datos CPU de Skroutz fue posible generar una red bayesiana con hasta 4 nodos padres. El tiempo de cómputo para 5 o más padres (variables con efecto en el nodo en cuestión) genera un espacio de búsqueda mayor que excedía 2 horas de cómputo en la computadora principal (MacBook) por lo cual se decidió mantener hasta 4 padres y revisar luego las distribuciones de los datos generados. La construcción de la red bayesiana se da mediante el análisis. El algoritmo GreedyBayes se utiliza para construir una red bayesiana a partir de un conjunto de datos (D). A continuación, se describe el funcionamiento de cada paso del algoritmo:

Símbolos usados:

$\mathcal{N}$: Conjunto de nodos y aristas de la red.

$\mathcal{V}$: Atributos seleccionados.

X_1: Primer atributo seleccionado del conjunto D.

$\emptyset$: Conjunto vacío.

A: Atributos del conjunto D.

Ω: Conjunto con las combinaciones de los nodos padres e hijos.

k: Número máximo de padres, es un valor especificado.

p: Valor mínimo entre k y $|\mathcal{V}|$.

Π: Cada posible conjunto de nodos padres.

D: Información mutua proveniente de los datos observados.

Inicialización: ($\mathcal{N}$) y ($\mathcal{V}$) se inicializan como conjuntos vacíos. ($\mathcal{N}$) almacenará las nodos y aristas de la red bayesiana, y ($\mathcal{V}$) almacenará los atributos ya seleccionados.

Selección inicial: Se selecciona aleatoriamente un atributo (X_1) del conjunto de atributos (A). Se añade (X_1,$\emptyset$) a ($\mathcal{N}$) y (X_1) a ($\mathcal{V}$). Esto significa que (X_1) no tiene padres iniciales.

Iteración sobre atributos restantes: Para cada atributo (X_1) (donde (i) varía de 2 a ($|\mathrm{A}|$) (cardinalidad del conjunto de atributos)):

> Se inicializa (Ω) como un conjunto vacío. (Ω) se utilizará para almacenar posibles combinaciones de nodos y sus padres.
>
> Se calcula (p) como el mínimo entre (k) (el número máximo de padres permitido) y ($|\mathcal{V}|$) (el número de atributos ya seleccionados).

Evaluación de posibles padres: Para cada atributo (X) en ($\mathrm{A} \setminus \mathcal{V}$) (atributos no seleccionados) y cada posible conjunto de padres (Π) de tamaño (p) en ($\mathcal{V}$):

> Se añade (X, Π) a (Ω).

Cálculo de información mutua:

> Se calcula la información mutua basada en (D) para todas las parejas en (Ω).

Selección de la mejor combinación: Se selecciona la combinación (X_i, Π_i) de (Ω) que maximiza la información mutua.

Actualización de la red: Se añade (X_i, Π_i) a ($\mathcal{N}$). Se añade (X_i) a ($\mathcal{V}$).

Finalización: El proceso se repite hasta que se hayan considerado todos los atributos en (A). Finalmente, se retorna ($\mathcal{N}$), que contiene la estructura de la red bayesiana construida.

Dependencias entre nodos de la red bayesiana:
A continuación, se presentan los nodos padres e hijos de la red bayesiana usada para generar los datos sintéticos y producto del algoritmo GreedyBayes.

Tabla 9 Red Neuronal con 5 nodos padres

Nodo	**Padres**
tfidf_distance	overlap_coefficient_distance
generalized_jaccard_distance	tfidf_distance, overlap_coefficient_distance
bag_distance	tfidf_distance, generalized_jaccard_distance, overlap_coefficient_distance
needleman_wunsch_distance	tfidf_distance, generalized_jaccard_distance,

	bag_distance_distance, overlap_coefficient_distance
affine_gap_distance	generalized_jaccard_distance, bag_distance_distance, needleman_wunsch_distance, overlap_coefficient_distance
levenshtein_distance	generalized_jaccard_distance, bag_distance_distance, needleman_wunsch_distance, overlap_coefficient_distance
smith_waterman_distance	generalized_jaccard_distance, bag_distance_distance, levenshtein_distance, overlap_coefficient_distance
partial_ration_distance	bag_distance_distance, levenshtein_distance, smith_waterman_distance, overlap_coefficient_distance
jaro_distance	bag_distance_distance, smith_waterman_distance, partial_ration_distance, generalized_jaccard_distance
jaro_winkler_distance	bag_distance_distance, levenshtein_distance, partial_ration_distance, overlap_coefficient_distance
soft_tfidf_distance	bag_distance_distance, partial_ration_distance, jaro_distance, generalized_jaccard_distance
overlap_coefficient_distance	Raíz

Generación de datos sintéticos usando la red bayesiana

Se generaron inicialmente 50.000 registros sintéticos y se comparan las distribuciones de cada variable, (gráfica izquierda representan los datos originales, la derecha son los datos sintéticos):

Levenshtein:

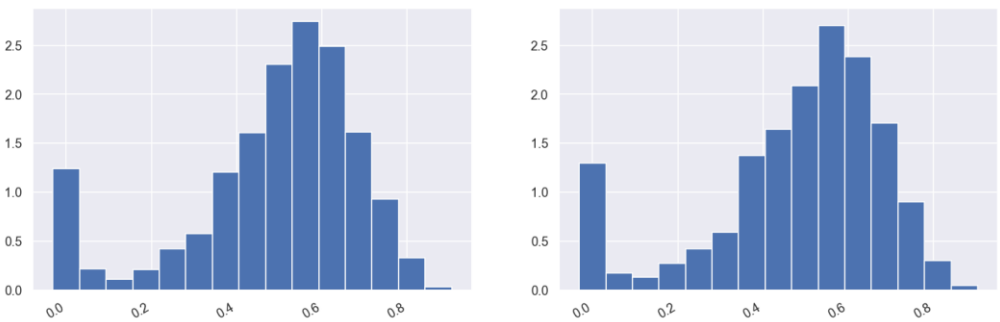


Ilustración 12 Distribuciones para la distancia de Levenshtein

Tabla 10 Estadísticos de las distribuciones para Levenshtein

Estadístico	Datos reales	Datos sintéticos
Mínimo	0	1.1267×10^{-05}
Máximo	0.9362	0.9352
Q1	0.4386	0.4271
Q2 (Mediana)	0.5647	0.5612
Q3	0.6571	0.6556
Promedio	0.5174	0.515
Moda	0.8000	No se determina

Needleman-Wunsch:

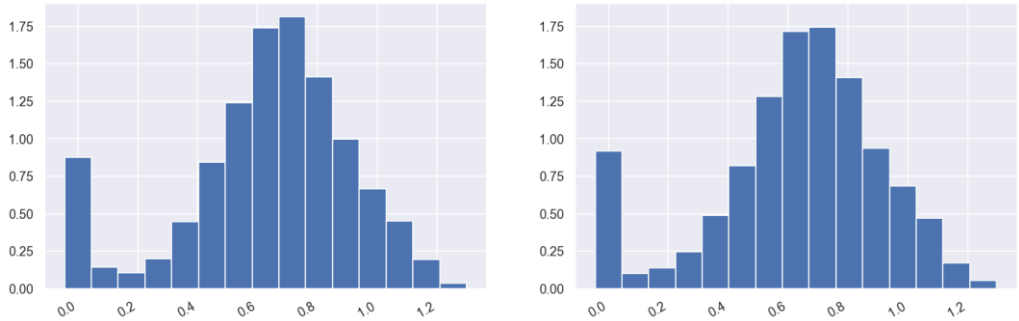


Ilustración 13 Distribuciones para la distancia de Needleman-Wunsh

Tabla 11 Estadísticos de las distribuciones para Needleman-Wunsh

Estadístico	Datos reales	Datos sintéticos
Mínimo	0	8.3416×10^{-05}
Máximo	1.3435	1.3416
Q1	0.5556	0.5421
Q2 (Mediana)	0.7160	0.7104
Q3	0.8605	0.8596
Promedio	0.6796	0.6758
Moda	0.7157	No se determina

Jaro:

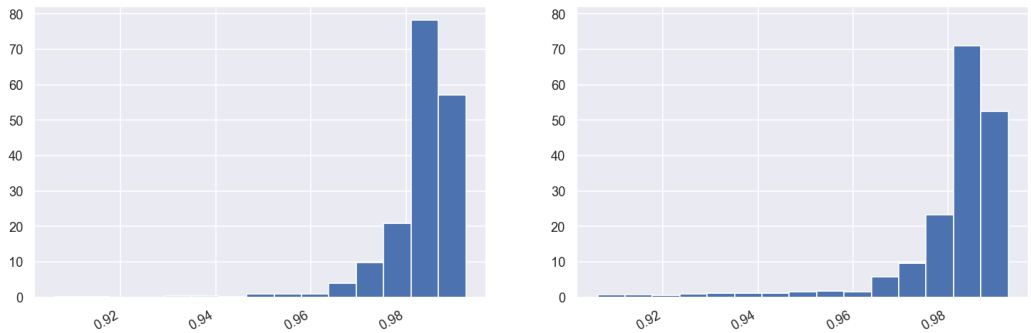


Ilustración 14 Distribuciones para la distancia de Jaro

Tabla 12 Estadísticos de las distribuciones para Jaro

Estadístico	Datos reales	Datos sintéticos
Mínimo	0.9091	0.9092
Máximo	0.9955	0.9955

Q1	0.9848	0.9835
Q2 (Mediana)	0.9881	0.9878
Q3	0.9904	0.9904
Promedio	0.9862	0.9847
Moda	0.8758	No se determina

Partial Ratio:

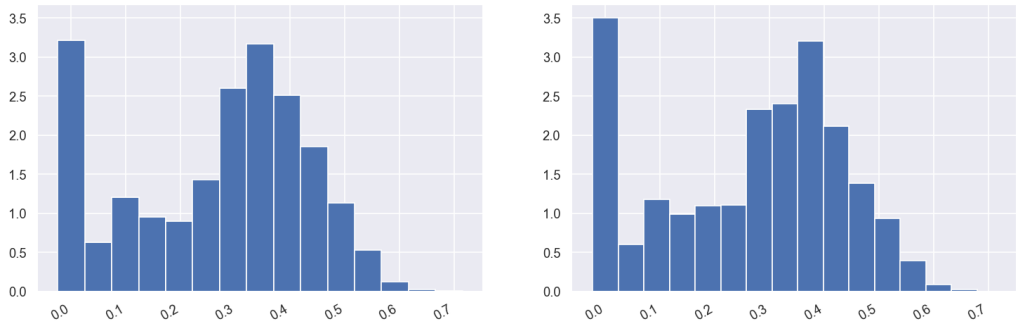


Ilustración 15 Distribuciones para la distancia para Partial Ratio

Tabla 13 Estadísticos de las distribuciones para Partial Ratio

Estadístico	Datos reales	Datos sintéticos
Mínimo	0	8.6548×10^{-06}
Máximo	0.7400	0.7388
Q1	0.1500	0.1462
Q2 (Mediana)	0.3300	0.3264
Q3	0.4200	0.4187
Promedio	0.2894	0.2922
Moda	0.6526	No se determina

Tanto los datos sintéticos generados por cópulas, como los generados por red bayesiana, no generan valores iguales, lo que no hace posible determinar la moda.

Se evidencia, en base a la información de las distribuciones de los datos sintéticos y reales, adicional a la información de las tablas de correlación, que los métodos de generación de datos por medio de cópulas y redes bayesianas son efectivos y puede ayudar a resolver los problemas relacionados con la falta de observaciones o registros reales.

- Salidas esperadas de la etapa metodológica: Versiones del modelo, informe de características, validación con los objetivos de la investigación.
- Ajuste metodológico: Interpretación de los modelos obtenidos.

Implementación o ajuste: En esta etapa se ajustaron los modelos y se realizaron tareas como validación cruzada, ajuste por hiperparámetros y se buscó obtener los modelos más precisos acorde a las métricas de clasificación.
Dado que la coincidencia de registros se aborda como un problema de clasificación, se emplean las métricas propias de un ejercicio de clasificación binaria:
Se parte del uso de la matriz de confusión con sus 4 cuadrantes:

TP: Verdaderos Positivos	FP: Falsos Positivos
FN: Falsos Negativos	TN: Verdaderos Negativos

Con la matriz se construyen las medidas:

Tabla 14 Métricas de clasificación

Métrica	Fórmula
Exactitud o Accuracy	$\frac{TP + TN}{TP + FN + FP + TN}$
Precisión	$\frac{TP}{TP + FP}$
Exhaustividad o Recall	$\frac{TP}{TP + FN}$
F1 Score	$\frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$

Adicional, se calculan sus variaciones:

Macro: Métrica para cada etiqueta usando la media sin tener presente el desbalanceo de clases.

Weighted: Se tiene en cuenta la cantidad de registros que hay para cada etiqueta y ayuda a manejar el desbalanceo de clases que hace ver mal a una métrica de una clase desbalanceada.

Salidas esperadas: Modelos más precisos e informe del ajuste realizado.

En el caso de las redes neuronales, también se realiza un ejercicio de ajuste de hiperparámetros en los escenarios posibles.

En la sección de Desarrollo y Resultados se expone los valores encontrados para cada modelo.

Implementación:

Arquitectura propuesta para un despliegue en producción: A continuación, se propone una arquitectura que permita aplicar y escalar los métodos bayesianos y consumir los resultados en una organización, sólo los componentes de Azure Databricks y cuentas de almacenamiento se desplegaron para el trabajo de producto cartesiano de las fuentes.

Es posible emplear herramientas de nube para alcanzar una paralelización en el procesamiento como también a la hora de incluir una funcionalidad de coincidencia de entidades en otros sistemas existentes. Se propone emplear los servicios de Azure como nube conocida por el autor para realizar el ejercicio de punta a punta desde conexión a fuentes estructuradas como bases de datos, a no estructuradas como archivos binarios o texto no estructurado, incluyendo información estructuradas que pueda venir de APIs en formatos como CSV, JSON o XML; movimiento de datos al ambiente nube, procesamiento masivo y en paralelo de preparación, calidad y características (cálculo por registros de

distancias); entrenamiento de modelos de aprendizaje y evaluación; y finalmente: consumo de resultados. A continuación, se detallan los componentes:

- Integration Runtime: Software de integración que corre en máquinas con acceso a las bases de datos privadas y fuentes no expuestas públicamente. Tiene acceso por medio de red a todos los sistemas heterogéneos necesarios.
- Data Factory: Servicio de orquestación y movimiento de datos. Puede escalar para adecuarse a volúmenes pequeños o Big Data, cuenta con gran cantidad de conectores como bases de datos y sistemas de ficheros para hacer el trabajo.
- Cuentas de almacenamiento: Almacenamiento escalable para mover los datos heterogéneos para luego procesar. Podría usarse para guardar resultados intermedios y binarios requeridos en el procesamiento.
- Databricks: Servicio de procesamiento en paralelo basado en Apache Spark, la idea es particionar los conjuntos de datos por lotes, donde se calculan las distancias de texto. Es posible también realizar entrenamientos en paralelo con Apache Spark y bibliotecas de cómputo distribuido.
- Machine Learning Service: Versionamiento de modelos y manejo de Web Services o Endpoints sobre Kubernetes dispuestos para escalar la evaluación de entidades para las aplicaciones en tiempo real.
- Kubernetes: Sistema de gestión de microservicios para exponer y escalar la evaluación en tiempo real del modelo de coincidencias.
- Power BI: Visualizador de resultados, podría usarse para validar el resultado de una tarea de búsqueda de coincidencia y métricas de entrenamiento.

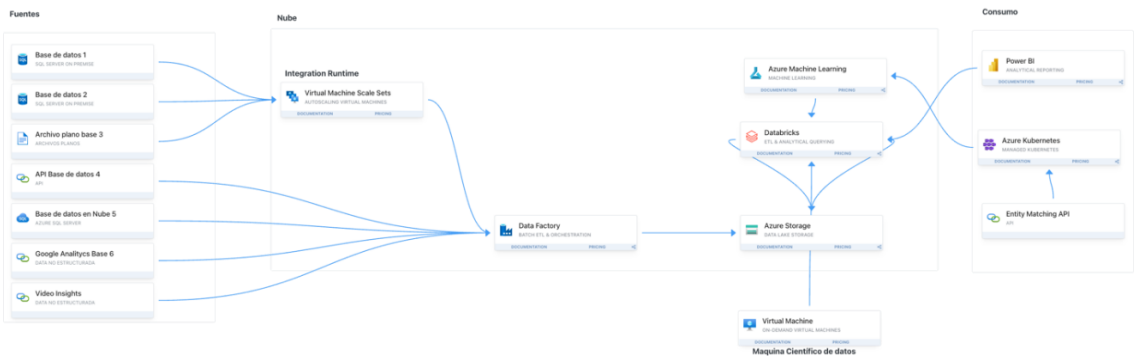


Ilustración 16 Arquitectura propuesta para una implementación en Producción

Aceptación del modelo:

La cual será una etapa para ver si los modelos alcanzaron el nivel esperado de predicción y para finalizar la documentación del proyecto.

- **Salidas esperadas**: análisis terminado, comparativa de métricas de clasificación, código fuente empleado para reproducir los experimentos.

Es importante mencionar que la fase de implementación cierra con el alcance de contar con cuadernillos que permiten replicar los experimentos realizados y gráficos comparativos de las diferentes técnicas probadas que se pueden consultar en el siguiente repositorio de Github.

DESARROLLO DE LOS MÉTODOS Y RESULTADOS

Se parte del subconjunto de CPUs de Skroutz, primero escalando los valores en un rango de 0 a 1 para cada característica o medida de distancia de texto, luego de tomar el 33% de los datos para probar y el resto para entrenar, se prueba con un primero modelo Naive Bayes Multinomial:
Matriz de confusión:

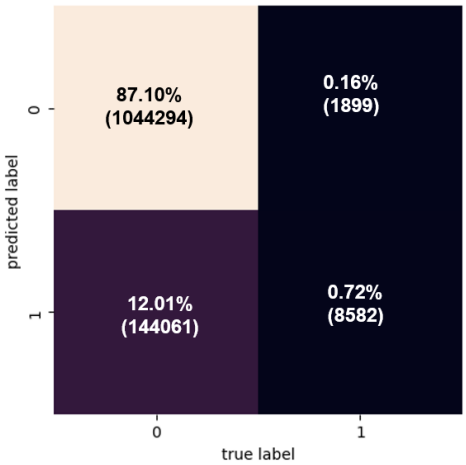


Ilustración 17 Matriz de confusión, modelo inicial

Métricas:

Tabla 15 Métricas del modelo inicial

	Precisión	**Recall**	**F1-Score**	**Cantidad registros**
No son iguales	1.00	0.88	0.93	1188355
Son iguales	0.06	0.82	0.11	10481
Exactitud			0.88	1198836
Macro	0.53	0.85	0.52	1198836
Weighted	0.99	0.88	0.93	1198836

El primer modelo resultado del primer entrenamiento se hizo con los datos desbalanceados: hay más registros que no coinciden de los que coinciden. La matriz de confusión muestra un posible problema para clasificar los registros coincidentes, pero estos valores bajos se deben también a los pocos registros que hay etiquetados como coincidentes, leyendo las métricas weighted de precisión, Recall y F1 se puede evidenciar una buena clasificación en ambas clases teniendo presente el desbalanceo.

En la siguiente iteración se procedió a complementar el conjunto de datos de entrenamiento de un millón de registros sintéticos, y búsqueda por hiperparámetros obteniendo los siguientes resultados luego de 204 modelos entrenados usando Validación Cruzada con 3 subconjuntos, el conjunto de datos de prueba no fue modificado, la métrica interna del entrenamiento es exactitud balanceada, que toma presente la cantidad de registros de cada clase:
Conjunto de entrenamiento:

- Registros que no coinciden: 2412719,
- Registros que coinciden: 1071281,
- Total registros: 3484000

Matriz de confusión, con los datos de prueba:

- Registros que no coinciden: 1188355,
- Registros que coinciden: 10481,
- Total registros: 1198836

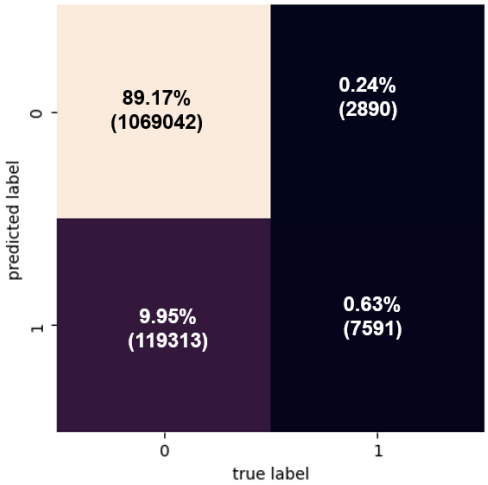


Ilustración 18 Matriz de confusión modelo Multinomial NB

Métricas:

Tabla 16 Métricas de clasificación para el modelo Multinomial NB

	Precisión	**Recall**	**F1-Score**	**Cantidad registros**
No son iguales	1.00	0.90	0.95	1188355
Son iguales	0.06	0.72	0.11	10481
Exactitud			0.90	1198836
Macro	0.53	0.81	0.53	1198836
Weighted	0.99	0.90	0.94	1198836

La prueba se repite con los modelos Naive Bayes Complemento y Bernoulli usando ajuste por hiperparámetros, validación cruzada con 3 subconjuntos y la inclusión de los datos sintéticos.
Cantidad de modelos entrenados Complemento y Bernoulli respectivamente: 102 y 1836.

Matrices de confusión: izquierda: Complemento, Bernoulli derecha

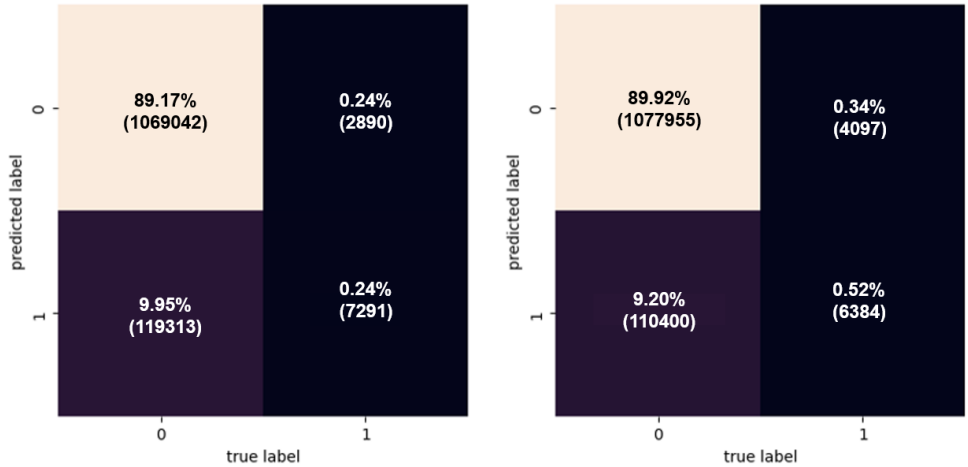


Ilustración 19 Matrices de confusión para Complemento y Bernoulli

Tablas de métricas:

Complemento:

Tabla 17 Métricas de clasificación modelo Complemento NB

	Precisión	**Recall**	**F1-Score**	**Cantidad registros**
No son iguales	1.00	0.90	0.95	1188355
Son iguales	0.06	0.72	0.11	10481
Exactitud			0.90	1198836
Macro	0.53	0.81	0.53	1198836
Weighted	0.99	0.90	0.94	1198836

Bernoulli:

Tabla 18 Métricas de clasificación modelo Bernoulli NB

	Precisión	**Recall**	**F1-Score**	**Cantidad registros**
No son iguales	1.00	0.91	0.95	1188355
Son iguales	0.05	0.61	0.10	10481
Exactitud			0.90	1198836
Macro	0.53	0.76	0.52	1198836
Weighted	0.99	0.90	0.94	1198836

Para comparar con técnicas del estado del arte, se propuso emplear generación de embebidos con los campos de cada producto y entrenamiento de redes neuronales de bibliotecas como Pytorch y Scikit learn.

Se generan versiones del conjunto de datos de Skroutz CPU con BERT, GLoVE, Word2Vec.

Con los embebidos generados, es posible aplicar medidas de distancia vectoriales, que suelen emplearse en arquitecturas como RAG (Generación aumentada de Recuperación en español). Dentro los componentes de búsqueda e indexado se incluyen medidas de distancia vectorial, al extraer tanto los vectores de audio, imágenes, videos, texto e información estructurada, es

posible recuperar la información relevante de forma más precisa y sin el requerimiento de unos metadatos completos. Dentro de las distancias implementadas se listan: Euclidiana, Coseno, Manhattan, Hamming, Jaccard y Pearson. Un primer ejercicio consistió en probar si las distancias vectoriales podrían usarse para hallar la coincidencia entre los productos.

Se procede a evaluar con diferentes valores frontera para separar los valores y se compara precisión y exhaustividad (Recall).

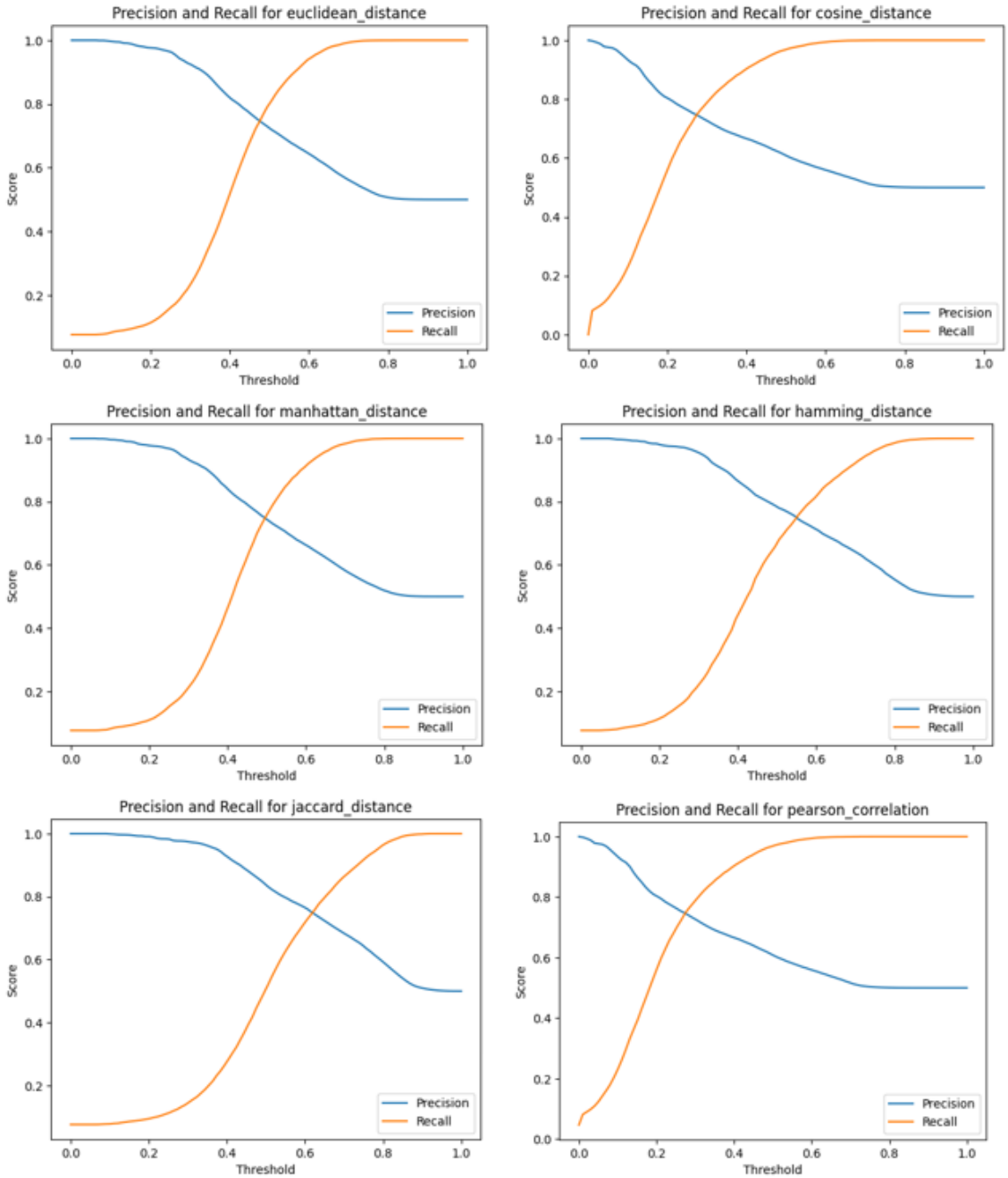


Ilustración 20 Distancias vectoriales como puntos de corte para clasificar las entidades

El punto óptimo para escoger se da en la intercepción de ambas líneas (precisión y Recall). A continuación, se resumen los valores de cada distancia.

Tabla 19 Punto óptimo de corte para las distancias vectoriales

Distancia	Precisión	Recall	Valor óptimo de corte
euclidean_distance	0.6312	0.9547	0.6162
cosine_distance	0.6245	0.9573	0.4747
manhattan_distance	0.6267	0.9564	0.6465
hamming_distance	0.6283	0.9377	0.7172
jaccard_distance	0.6256	0.9351	0.7677
pearson_correlation	0.6302	0.9513	0.4646

La siguiente prueba consistió en entrenar una red neuronal con 3 capas:

1. Lineal con entrada de 600 parámetros (usando word2vec se generaron vectores con 600 valores de las dos descripciones de los productos a comparar), 5000 parámetros de salida. Función de activación tipo relu.
2. Lineal de 5000 parámetros de entrada y 5000 de salida. Función de activación tipo relu.
3. Lineal de 5000 parámetros de entrada y 1 salida. Función de activación sigmoide.

Durante el entrenamiento se visualiza una convergencia en base a la pérdida (loss):

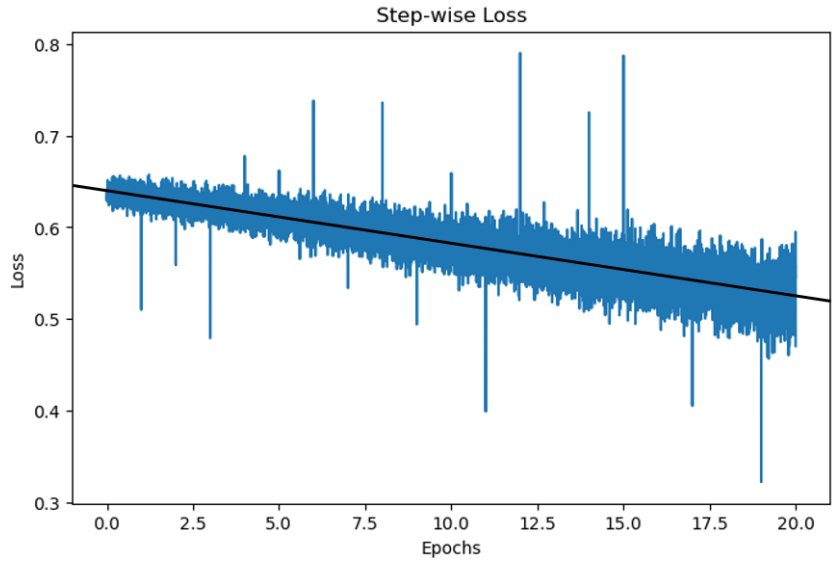


Ilustración 21 Convergencia red neuronal inicial

Al evaluar con los datos de pruebas del mismo conjunto se lograron resultados aceptables:

Tabla 20 Métricas de clasificación de la red neuronal inicial

	Precisión	Recall	F1-Score	Cantidad registros
No son iguales	0.81	0.87	0.84	10482
Son iguales	0.86	0.80	0.83	10481
Exactitud			0.84	20963
Macro	0.84	0.84	0.84	20963
Weighted	0.84	0.84	0.84	20963

Al probar contra los conjuntos de datos de Skroutz Tv y PriceRunner cámaras, los resultados revelan que el modelo no alcanza a generalizar bien y tiene problemas para identificar cuando los modelos no coinciden:

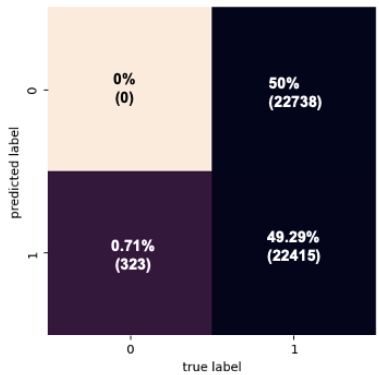


Ilustración 22 Matriz de confusión de la red neuronal con los datos alternos

Tabla 21 Métricas de clasificación red neuronal con los datos alternos

	Precisión	**Recall**	**F1-Score**	**Cantidad registros**
No son iguales	0.00	0.00	0.00	22738
Son iguales	0.50	0.99	0.66	22738
Exactitud			0.49	45476
Macro	0.25	0.49	0.33	45476
Weighted	0.25	0.49	0.33	45476

Cámaras:

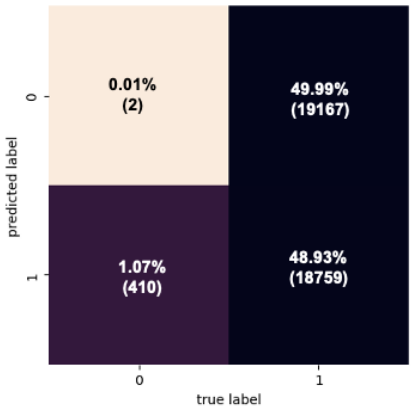


Ilustración 23 Matriz de confusión de la red neuronal con los datos alternos 2

Tabla 22 Métricas de clasificación red neuronal con los datos alternos

	Precisión	**Recall**	**F1-Score**	**Cantidad registros**
No son iguales	0.00	0.00	0.00	19169
Son iguales	0.49	0.98	0.66	19169
Exactitud			0.49	38338
Macro	0.25	0.49	0.33	38338
Weighted	0.25	0.49	0.33	38338

Se intentó realizar una prueba con una red neuronal recurrente (RNN), que acorde al estado del arte, son muy utilizadas para tareas de tipo NLP y se desempeñan bien con embebidos. Las capas utilizadas son:

1. Capa lineal, 600 características de entrada, salida a 600 características.
2. Capa escondida de 600 de entrada por 600 de salida.
3. Capa final de 600 de entrada por una salida, softmax usado en esta capa.

Luego de múltiples intentos de configuración y variaciones, no fue posible llegar a un modelo con convergencia:

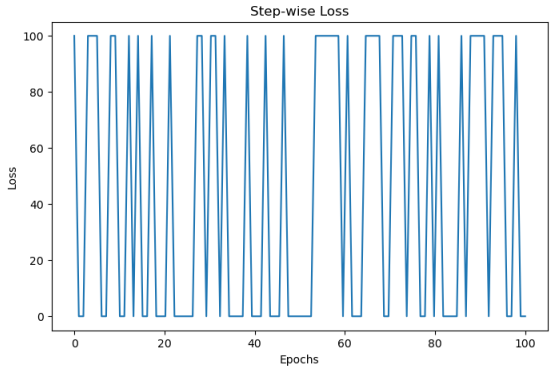


Ilustración 24 Convergencia de la red neuronal RNN

Las métricas demuestran que no se logra un modelo que pueda superar un lanzamiento aleatorio de moneda:

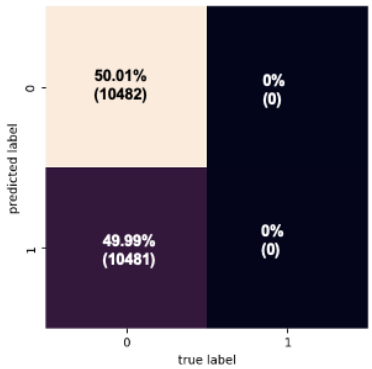


Ilustración 25 Matriz de confusión red neuronal RNN

Tabla 23 Métricas de clasificación red neuronal RNN

	Precisión	**Recall**	**F1-Score**	**Cantidad registros**
No son iguales	0.50	1.00	0.67	10482
Son iguales	0.00	0.00	0.00	10481
Exactitud			0.50	20963
Macro	0.25	0.50	0.33	20963
Weighted	0.25	0.50	0.33	20963

La última prueba realizada, consistió en probar con el algoritmo de máquinas de soporte Vectorial, propiamente SVM lineal.

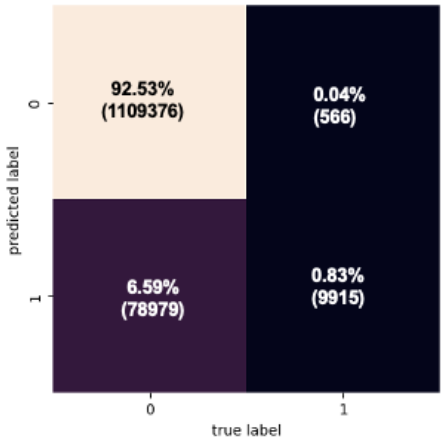


Ilustración 26 Matriz de confusión modelo SVM lineal

Tabla 24 Métricas de clasificación modelo SVM Lineal

	Precisión	**Recall**	**F1-Score**	**Cantidad registros**
No son iguales	1.00	0.93	0.97	1188355
Son iguales	0.11	0.95	0.20	10481
Exactitud			0.93	1198836
Macro	0.25	0.50	0.33	1198836
Weighted	0.25	0.50	0.33	1198836

CONCLUSIONES

Discusión de los resultados

En este trabajo se abordaron varias técnicas estadísticas combinadas con aprendizaje de máquina para abordar el problema de la coincidencia de entidades en bases de datos heterogéneas. A continuación, se presentan los resultados de los modelos convergentes y sus estadísticas calculadas teniendo presente el desbalanceo en el conjunto de pruebas:

Tabla 25 Resúmen de métricas de clasificación para los modelos probados

Modelo	Precisión	Recall	F1-Score	Cantidad registros de prueba
Naive Bayes Multinomial	0.99	0.88	0.93	1198836
Naive Bayes Multinomial con registros sintéticos	0.99	0.90	0.94	1198836
Complemento NB	0.99	0.90	0.94	1198836
Bernoulli NB	0.99	0.90	0.94	1198836
Red neuronal de 3 capas	0.84	0.84	0.84	20963
RNN	0.25	0.50	0.33	20963
SVM Lineal	0.25	0.50	0.33	1198836

Los modelos Multinomial, Complemento, Bernoulli alcanzaron resultados similares de clasificación, hubo una leve mejora al emplear datos sintéticos.

Uno de los retos para abordar la coincidencia de entidades radica en la poca cantidad de registros coincidentes con los que se pueda contar, se abordó este problema con la generación de datos sintéticos. Tanto los datos sintéticos generados por cópulas como los generados por redes bayesianas lograron distribuciones similares, lo que permite emplear los registros sintéticos para mejorar los modelos y reducir los problemas asociados al desbalanceo de clases.

Las características basadas en funciones de distancia de texto presentaron varias ventajas frente a otras representaciones como los embebidos. Las funciones de texto pueden ser calculadas de forma independiente una de las otras, esto permite paralelizar el procesamiento. El cálculo de embebidos requiere de otros pasos secuenciales como definiciones de corpus y uso de redes neuronales para representar el texto.

Los métodos de aprendizaje automático basados en Bayes presentan ventajas interesantes sobre métodos del estado del arte para la coincidencia de entidades:

- Convergencia a un modelo con menor cantidad de recursos computacionales.
- Variedad en modelos para abordar un problema de coincidencia cuando los datos empleados pueden ser: discretos, categóricos, continuos o de naturaleza binaria.
- Naturaleza de modelo tipo caja blanca: Es posible interpretar los modelos y sus valores.

Las medidas de distancia entre textos, conjuntos de datos y representación vectorial de textos, pueden ser usados como características para representar las entidades. Es necesario evaluar el dominio del problema a abordar, y las restricciones de recursos (como tiempo de procesamiento o memoria computacional) para escoger la mejor representación de las características:

- Tiempo requerido para generar los embebidos: 55 minutos, registros: 20963.
- Tiempo requerido para generar las distancias de texto: 22 minutos, registros: 1198836.

Las técnicas bayesianas demostraron ser efectivas para la coincidencia de entidades en conjuntos de datos grandes. Estas técnicas permiten manejar la incertidumbre y la variabilidad inherente en los datos mediante la formulación de probabilidades condicionales y la integración de conocimiento experto.

Trabajo futuro

Aunque los métodos bayesianos son prometedores, existen desafíos pendientes, como la necesidad de grandes volúmenes de datos etiquetados para el entrenamiento y la validación de modelos. Futuras investigaciones podrían enfocarse en desarrollar técnicas de aprendizaje automático que requieran menos datos etiquetados y en la integración de métodos bayesianos con otras técnicas de aprendizaje profundo.

Las futuras investigaciones pueden abordar la aplicación de técnicas estadísticas bayesianas en la coincidencia de entidades utilizando conjuntos de datos más diversos y heterogéneos. Específicamente, se podrían explorar bases de datos provenientes de diferentes dominios, como registros médicos, transacciones financieras, y datos de redes sociales, para evaluar la eficacia y robustez de los métodos bayesianos en contextos variados.

REFERENCIAS

Akritidis, L., & Bozanis, P. (2018). Effective Unsupervised Matching of Product Titles with k-Combinations and Permutations. *Proceedings of the 14th IEEE International Conference on Innovations in Intelligent Systems and Applications (INISTA)*, 1-10.

Akritidis, L., Fevgas, A., Bozanis, P., & Makris, C. (2020). A self-verifying clustering approach to unsupervised matching of product titles. *Artificial Intelligence Review*, 53, 4777-4820.

Barlaug, N., & Gulla, J. (2021). Neural networks for entity matching: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1-37.

Brunner, U., & Stockinger, K. (2020). Entity matching with transformer architectures-a step forward in data integration. *23rd International Conference on Extending Database Technology* (págs. 463-473). Copenhagen: OpenProceedings.

Corporation, M. (21 de febrero de 2024). *TDSP*. Obtenido de Microsoft Docs: https://docs.microsoft.com/es-es/azure/architecture/data-science-process/overview

Doan, A., Ives, Z., & Halevy, A. (2012). *Principles of data integration.* Elsevier.

Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., & Tang, N. (2018). Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, 1454–1467.

Echeverri Calderón, S. (2022). *Metodología para el análisis de la similitud entre marcas mediante técnicas de aprendizaje automático.* Medellín, Colombia: Universidad EAFIT.

IBM. (25 de Abril de 2024). *IBM Sitio Oficial*. Obtenido de ¿Qué son las redes neuronales recurrentes?: https://www.ibm.com/es-es/topics/recurrent-neural-networks

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques.* MIT press.

Konda, P., Das, S., Suganthan, P., Doan, A., Ardalan, A., Ballard, J. R., . . . Raghavendra, V. (2018). *Magellan: Toward Building Entity Matching Management Systems.* University of Wisconsin-Madison.

Laurenza, E. (2015). Solving conflicts in database fusion with Bayesian networks. *International Conference on Information Fusion (Fusion)* (págs. 399-406). Washington, DC: IEEE.

Marchant, N. G., Kaplan, A., Elazar, D. N., Rubinstein, B. I., & Steorts, R. C. (2021). d-blink: Distributed end-to-end Bayesian entity resolution. *Journal of Computational and Graphical Statistics*, 406-421.

Marshall, J. T. (1947). Canada's national vital statistics index. *Population Studies*, 204-211.

Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., & Raghavendra, V. (2018). Deep Learning for Entity Matching: A Design Space Exploration. *Proceedings of the 2018 international conference on management of data*, 19-34.

Newcombe, B. H., Kennedy, M. J., Axford, S. J., & James, A. P. (1959). Automatic linkage of vital records. *American Association for the Advancement of Science*, 954-959.

Paganelli, M., Del Buono, F., Guerra, F., Pevarello, M., & Vincini, M. (2021). Automated Machine Learning for Entity Matching Tasks. *Open Proceedings*, 325-330.

Ping, H., Stoyanovich, J., & Howe, B. (2017). Datasynthesizer: Privacy-preserving synthetic datasets. *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, 1-5.
Ramezani, M., Ilangovan, G., & Kum, H. (2021). Evaluation of machine learning algorithms in a human-computer hybrid record linkage system. *CEUR workshop proceedings*, Vol. 2846, No. 4.
Ramezani, M., Ilangovan, G., & Kum, H.-C. (2021). Evaluation of Machine Learning Algorithms in a Human-Computer Hybrid Record Linkage System. *CEUR workshop proceedings, 2846*.
Restrepo, J., Rivera, J., Laniado, H., Osorio, P., & Becerra, O. (2023). Nonparametric Generation of Synthetic Data Using Copulas. *Electronics*, 12(7), 1601.
Shearer, C. (2000). The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, 13-22.