1

**UNIVERSIDAD AUTÓNOMA DE MADRID**

**ESCUELA POLITÉCNICA SUPERIOR**

**Gráfico de barras

Descripción generada automáticamente con confianza media**

**Grado en Ingeniería Informática**

**TRABAJO FIN DE GRADO**

**Análisis y Estrategias para la Mitigación del Phishing**

**Un Enfoque en la Ciberseguridad**

**Autor: Adrián Crespo Musheghyán**

**Tutor: Óscar Delgado**

**Junio 2024**

### Agradecimientos

### Resumen

Resumen

**Palabras clave**

Palabras clave

### Abstract

Abstract

**Keywords**

Keywords

### Índice

[Agradecimientos III](#_Toc127720433)

[Resumen V](#_Toc127720434)

[Abstract VII](#_Toc127720435)

[Índice de figuras XI](#_Toc127720437)

[Índice de tablas XI](#_Toc127720438)

[Índice de cuadros XI](#_Toc127720439)

[1 Introducción 1](#_Toc127720440)

[1.1. Motivación 1](#_Toc127720441)

[1.2. Objetivos 1](#_Toc127720442)

[2 Estado del arte 3](#_Toc127720443)

[2.1. Subsección 3](#_Toc127720444)

[2.2. Subsección 3](#_Toc127720445)

[2.3. Subsección 3](#_Toc127720446)

[2.3.1. Subsubsección 3](#_Toc127720447)

[Bibliografía 7](#_Toc127720448)

[Apéndices 9](#_Toc127720449)

[Apéndice A 11](#_Toc127720450)

### Índice de figuras

X.X Figura 1 XX

### Índice de tablas

X.X Tabla 1 XX

### Índice de cuadros

X.X Cuadro 1 XX

# Introducción

#### Motivación

Motivación

#### Objetivos

Objetivos:

* Objetivo1
* Objetivo2

# AAlgoritmos de comparación de cadenas

En el mundo de la informática, la capacidad de comprender y procesar el lenguaje humano es una tarea fundamental y compleja. Aunque para los humanos puede resultar intuitivo reconocer y corregir errores tipográficos o malas ortografías en el texto, para las máquinas esta tarea puede resultar considerablemente más desafiante. La concordancia difusa de cadenas, también conocida como "fuzzy string matching", emerge como una técnica esencial para abordar este desafío al permitir la identificación de cadenas de texto que coinciden parcialmente pero no exactamente.

La distancia entre cadenas se basa en el reconocimiento de patrones y la comparación inteligente de cadenas de texto para encontrar similitudes incluso en presencia de errores tipográficos, malas ortografías o variaciones leves en las palabras. Esto es especialmente prominente en los motores de búsqueda, donde los usuarios a menudo introducen consultas con errores ortográficos o variantes de palabras.

Por ejemplo, si un usuario escribiera "Londin" en lugar de "London" en Google, la concordancia difusa de cadenas identificaría que "London" era la palabra buscada, y Google devolvería resultados de búsqueda para ella.

Sin embargo, la distancia entre cadenas va mucho más allá de la mera corrección ortográfica en los motores de búsqueda. Esta técnica tiene aplicaciones en una variedad de campos, desde la corrección automática de texto en procesadores de texto hasta la identificación de nombres de productos en bases de datos comerciales. En el ámbito médico, por ejemplo, la distancia entre cadenas puede ayudar a identificar variantes ortográficas de nombres de medicamentos o enfermedades en registros electrónicos de salud, facilitando así la búsqueda y el análisis de datos clínicos.

Además, en el ámbito de la seguridad informática, la concordancia difusa de cadenas tiene una gran importancia en la detección de fraudes y el análisis forense de datos.

Los investigadores utilizan esta técnica para identificar patrones sospechosos en correos electrónicos, mensajes de texto o registros de actividad en línea, así como para detectar Phishing, lo que les permite detectar posibles amenazas cibernéticas y tomar medidas preventivas.

#### Distancia de Hamming

La Distancia de Hamming es un concepto fundamental en el campo de la teoría de la información y la codificación de datos. Su historia se remonta a la década de 1950, cuando el matemático estadounidense Richard Hamming desarrolló este concepto como parte de su trabajo en el diseño de códigos de corrección de errores para sistemas de comunicación digital. Richard Hamming fue pionero en la aplicación de técnicas matemáticas en el campo emergente de la informática y las telecomunicaciones. Durante su tiempo en los Laboratorios Bell de Investigación en Informática (Bell Labs), Hamming trabajó en una variedad de proyectos relacionados con la codificación de datos y la detección y corrección de errores en sistemas de transmisión digital. En teoría de la información se denomina distancia de Hamming a la efectividad de los códigos de bloque y depende de la diferencia entre una palabra de código válida y otra. Cuanto mayor sea esta diferencia, menor es la posibilidad de que un código válido se transforme en otro código válido por una serie de errores. A esta diferencia se le llama distancia de Hamming, y se define como el número de bits que tienen que cambiarse para transformar una palabra de código válida en otra palabra de código válida.

Si dos palabras de código difieren en una distancia d, se necesitan d errores para convertir una en la otra.

Por ejemplo:

La distancia Hamming entre 1011101 y 1001001 es 2.

La distancia Hamming entre 2143896 y 2233796 es 3.

La distancia Hamming entre "tener" y "reses" es 3.

El funcionamiento del algoritmo de la Distancia de Hamming es directo. Para calcular la distancia entre dos cadenas, simplemente se comparan carácter por carácter y se cuentan las posiciones en las que difieren. Este enfoque es especialmente útil en aplicaciones donde las cadenas representan datos binarios, como en la transmisión de información digital o la detección de errores en sistemas de almacenamiento. La Distancia de Hamming ha encontrado aplicaciones en una variedad de campos, incluidos los códigos de corrección de errores, la bioinformática, la criptografía y la teoría de la complejidad computacional. También, se emplea para el uso de detección de Phising, nos permite cuantificar la similitud entre dos urls, La distancia de Hamming es un algoritmo con una base simple pero muy potente ya que nos permite detectar urls potenciales de phishing hacia una entidad. El principal inconveniente es que llas dos cadenas tienen que tener una misma longitud para que el algoritmo sea efectivo por lo que si un atacan suplanta a gruposantander.com con la url grupsantander.com, la distancia de haming no devolvería un análisis lo suficientemente completo com para detectar Phising.

A computer screen with text

Description automatically generated

#### Distancia de Levenshtein

La distancia de Levenshtein, distancia de edición o distancia entre palabras es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra, se usa ampliamente en teoría de la información y ciencias de la computación. Se entiende por operación, bien una inserción, eliminación o la sustitución de un carácter. Esta distancia recibe ese nombre en honor al científico ruso Vladimir Levenshtein, quien se ocupó de esta distancia en 1965. Es útil en programas que determinan cuán similares son dos cadenas de caracteres, como es el caso de los correctores ortográficos. Por ejemplo, la distancia de Levenshtein entre "casa" y "calle" es de 3 porque se necesitan al menos tres ediciones elementales para cambiar uno en el otro. casa → cala (sustitución de 's' por 'l') cala → calla (inserción de 'l' entre 'l' y 'a') calla → calle (sustitución de 'a' por 'e') Se le considera una generalización de la distancia de Hamming, que solo compara cadenas de la misma longitud y solo considera como operación la sustitución. Hay otras generalizaciones de la distancia de Levenshtein, como la distancia de Damerau-Levenshtein, que consideran el intercambio de dos caracteres como una operación. La distancia de Levenshtein tiene una amplia variedad de aplicaciones prácticas. Por ejemplo, en la corrección ortográfica, se utiliza para sugerir palabras correctas cuando una palabra está mal escrita. En la bioinformática, se emplea para comparar secuencias genéticas y determinar su similitud o parentesco evolutivo. En la detección de plagio, se utiliza para identificar similitudes entre textos y detectar posibles casos de copia no autorizada. Este algoritmo, tiene más alcance que la distancia de Hamming ya que nos permie comparar distancias entre cadenas de distinta longitud. El resultado del algoritmo, potemos cuantificarlo en un valor del 0 a 1 siendo 0 totalmente distinto y 1 una copia idéntica de la siguiente forma:

max\_len = max(len(linea), len(url))

levenshtein\_dist = levenshtein\_distance(linea, url)

similarity = 1 - (levenshtein\_dist / max\_len)

Siendo *similarity*, el valor entre 0 y 1 que nos cuantifica la similitud entre ambas cadenas.

A screen shot of a computer code

Description automatically generated

#### Distancia de Damerau Levenshtein

La distancia de Damerau-Levenshtein es una variante de la distancia de Levenshtein que también tiene en cuenta las transposiciones de caracteres además de las operaciones de inserción, eliminación y sustitución. Esta métrica lleva el nombre de los científicos soviéticos Vladimir Damerau y Vladimir Levenshtein, quienes contribuyeron significativamente al desarrollo de la teoría de la información y la informática en la década de 1960. La distancia de Damerau-Levenshtein se utiliza para medir la similitud entre dos cadenas de caracteres al calcular la cantidad mínima de operaciones requeridas para convertir una cadena en la otra, incluyendo transposiciones adyacentes de caracteres. Esto la hace especialmente útil en aplicaciones donde las transposiciones son comunes, como la corrección ortográfica y la comparación de palabras. El algoritmo de Damerau-Levenshtein es una extensión del algoritmo de Levenshtein y utiliza una matriz similar para calcular la distancia entre las cadenas. Sin embargo, también considera las transposiciones de caracteres al determinar el camino óptimo a través de la matriz. Esto permite una mayor precisión en la medición de la similitud entre las cadenas y una mejor detección de errores ortográficos o de escritura. La distancia de Damerau-Levenshtein tiene una amplia variedad de aplicaciones en campos como la corrección ortográfica, la bioinformática, la detección de plagio y la comparación de textos. Su capacidad para tener en cuenta las transposiciones de caracteres la hace especialmente útil en situaciones donde las cadenas pueden contener errores tipográficos o variaciones leves en la ortografía. Con lo que esta versión mejorada del algoritmo anterior de Levenhstein es más precisa a la hora de detectar Phising en dos cadenas (urls). Aplicamos un razonomiento similar al anterior para cuentificar la similiritud entre dos cadenas y así, sacar un valor que nos permita detectar casos de phisings.

A computer screen shot of a program

Description automatically generated

#### Índice de Jaccard

El ***índice de Jaccard ( IJ )*** o ***coeficiente de Jaccard ( IJ )*** mide el grado de similitud entre dos conjuntos, sea cual sea el tipo de elementos.

La formulación es la siguiente:

A mathematical equation with black text

Description automatically generated

Es decir, la cardinalidad de la intersección de ambos conjuntos dividida por la cardinalidad de su unión.

Siempre toma valores entre 0 y 1, correspondiente este último a la igualdad total entre ambos conjuntos.

En [ecología](https://es.wikipedia.org/wiki/Ecolog%C3%ADa) se usa para medir la similitud, disimilitud o distancias (estas dos últimas si se le resta el índice a 1) que existen entre dos estaciones de muestreo, con una formulación equivalente:[1](https://es.wikipedia.org/wiki/%C3%8Dndice_de_Jaccard#cite_note-1)​**IJ: c / (a+b-c)**

A mathematical equation with black text

Description automatically generated

Donde:

* **a:** es el número de especies presentes en la estación A.
* **b:** es el número de especies presentes en la estación B.
* **c:** es el número de especies presentes en ambas estaciones, A y B.

En [informática](https://es.wikipedia.org/wiki/Inform%C3%A1tica) se utiliza para medir la distancia entre vectores definidos sobre un espacio vectorial booleano (las componentes del vector sólo pueden ser 0 o 1).

**J(A,B) = |A ∧ B| / |A ∨ B|**

donde ∧ y ∨ son, respectivamente, las operaciones × (AND) y + (OR) de la lógica booleana, y |A|=∑ai.

En el caso de URLs, con el objetivo de detectar phishing, podemos considerar cada URL como un conjunto de tokens, como palabras o caracteres, dependiendo de la granularidad deseada.

Para utilizar el algoritmo de Jaccard en la detección de phishing de URLs, primero representamos cada URL como un conjunto de tokens. Luego, calculamos la similitud de Jaccard entre la URL sospechosa y una lista de URLs conocidas como legítimas. Si la similitud de Jaccard entre la URL sospechosa y alguna de las URLs legítimas no supera un umbral predefinido, podemos considerar que la URL sospechosa tiene una alta probabilidad de ser legítima. De lo contrario, podría tratarse de un intento de phishing.

#### A computer screen with white and orange text Description automatically generated

#### Distancia de Cosine

La similitud del coseno es una métrica que determina en qué se parecen dos vectores (palabras, oraciones, características) entre sí. Básicamente, es un ángulo entre dos vectores.



El rango de similitud está entre -1 y 1, donde -1 vectores absolutamente opuestos (python - seguridad del código) , 0 sin correlación (conocimiento universitario - trabajo) , 1 absolutamente similar (chatgpt - hype) . Se puede explicar nuevamente por los ángulos, recuerda el coseno de dos vectores que apuntan en direcciones diferentes. El ángulo entre ellos es 180, son opuestos, el coseno es igual a -1. También, se puede plantear la utilización de la distancia del coseno, que se define como 1 – similitud del coseno. Para detectar phishing y calcular similitud en las urls, cualquiera de las dos sería válida ya que el razonamiento para calcular el porcentaje de similitud sería el mismo, pero a la inversa.

A screen shot of a computer code

Description automatically generated

#### Distancia de Jaro Winkler

La distancia de Levenshtein es una medida que cuantifica el número de operaciones requeridas para transformar una cadena de caracteres en otra. Sin embargo, otra métrica ampliamente utilizada para evaluar la similitud entre dos cadenas de texto es la distancia de Jaro-Winkler. Esta métrica, propuesta en 1990 por Winkler como una modificación de la distancia de Jaro desarrollada un año antes, otorga un peso adicional a las cadenas que comparten prefijos. A diferencia de la distancia de Levenshtein, el algoritmo de Jaro-Winkler proporciona directamente un valor de similitud entre 0 y 1, lo que elimina la necesidad de normalizar posteriormente los resultados. Esta característica hace que el algoritmo de Jaro-Winkler sea particularmente útil en aplicaciones donde se necesita una evaluación rápida y precisa de la similitud entre cadenas de texto. Al considerar las similitudes entre cadenas de texto, tanto la distancia de Levenshtein como la distancia de Jaro-Winkler ofrecen herramientas poderosas y complementarias. Mientras que la distancia de Levenshtein se centra en la cantidad de operaciones necesarias para igualar las cadenas, la distancia de Jaro-Winkler destaca la importancia de los prefijos compartidos y proporciona una medida más intuitiva de la similitud entre las cadenas.

#### A screen shot of a computer program Description automatically generated

#### Conclusión

Tras analizar los distintos algoritmos planteados para cuantificar la similitud entre dos cadenas como urls para detectar phishing se han encontrado los siguientes pros y contras para cada algoritmo expuesto:

**1. Distancia de Hamming:**

Pros:

Simplicidad: La distancia de Hamming es fácil de entender y calcular.

Eficiencia: Es eficiente en la comparación de cadenas de igual longitud, lo que la convierte en una opción rápida para detectar discrepancias en URLs de longitud fija.

Contras:

Longitud fija de las cadenas: Limitada a cadenas de igual longitud.

Sensibilidad a transposiciones: No detecta transposiciones de caracteres, lo que podría pasar por alto casos de phishing donde los atacantes modifican el orden de los caracteres en las URLs.

**2. Distancia de Levenshtein:**

Pros:

Flexibilidad: Puede manejar cadenas de diferentes longitudes.

Sensibilidad a errores: Es capaz de detectar una amplia gama de cambios entre cadenas.

Contras:

Costo computacional: El cálculo de la distancia de Levenshtein es computacionalmente más costoso.

Sensibilidad a transposiciones: No detecta transposiciones de caracteres, lo que podría pasar por alto casos de phishing donde los atacantes modifican el orden de los caracteres en las URLs.

**3. Distancia de Levenshtein Damerau:**

Pros:

Manejo de transposiciones: A diferencia de la distancia de Levenshtein estándar, tiene en cuenta las transposiciones de caracteres.

Sensibilidad a errores: Al igual que la distancia de Levenshtein, puede detectar inserciones, eliminaciones, sustituciones y transposiciones de caracteres.

Contras:

Costo computacional: Similar a la distancia de Levenshtein, puede ser computacionalmente costoso.

Sensibilidad a la longitud de las cadenas: A medida que aumenta la longitud de las cadenas, el costo computacional también aumenta, lo que puede limitar su aplicabilidad en la detección de phishing en URLs largas.

**4. Algoritmo de Jaccard:**

Pros:

Simplicidad: Calcula la similitud entre conjuntos de elementos.

Flexibilidad: No requiere que las cadenas tengan la misma longitud.

Contras:

Sensibilidad al contenido del conjunto: La similitud de Jaccard puede ser influenciada por el contenido específico del conjunto de elementos.

Limitación a conjuntos: Se basa en la representación de las cadenas como conjuntos de elementos, lo que puede limitar su aplicabilidad en la detección de phishing en URLs que requieren una comparación más precisa de los caracteres individuales.

**5. Algoritmo de Cosine:**

Pros:

Representación vectorial: Calcula la similitud entre dos vectores.

Flexibilidad: No requiere que las cadenas tengan la misma longitud.

Contras:

Sensibilidad a la representación vectorial: La calidad de la representación vectorial puede afectar la precisión del algoritmo, especialmente en la detección de phishing en URLs donde la representación de términos puede variar.

Sensibilidad a la longitud de las cadenas: A medida que aumenta la longitud de las cadenas, el espacio vectorial también puede aumentar, lo que puede afectar la eficiencia computacional en el cálculo de la similitud.

**6. Algoritmo de Jaro-Winkler:**

Pros:

Consideración de prefijos compartidos: Otorga un mayor peso a los prefijos compartidos entre las cadenas, lo que puede mejorar la precisión en la detección de similitudes en URLs con prefijos similares.

Valor de similitud directa: Proporciona un valor de similitud entre 0 y 1, lo que facilita la interpretación de los resultados.

Contras:

Sensibilidad a la longitud de las cadenas: Puede ser menos efectivo en la comparación de URLs de diferentes longitudes.

Limitación a cadenas de texto: Se basa en la comparación de caracteres individuales en las cadenas de texto, lo que puede limitar su aplicabilidad en la detección de phishing en URLs que requieren una comparación más precisa de los componentes de la URL.

Por lo que el algoritmo de los anteriormente expuestos con sus pros y cntras, el que mejor se ajusta a la búsqueda de urls potenciales de phishing a partir de la url original es el de **Jaro-Winkler.** Por ejemplo, en una muestra de las últimas urls creadas a día 2 de mayo de 2024, con la palabra **facebook.com** comourl legítima nos avisa de las siguientes urls con un umbral de similitud establecido al 80%:

Jaro-Winkler distance between facebookagi.com

and facebook.com is 0.98 (98.33%)

Jaro-Winkler distance between facerbook.in

and facebook.com is 0.88 (88.85%)

Observamos grandes parecidos entre ambos dominós, lo cual puede llevar a confusión a la víctima.

### Bibliografía

|  |  |
| --- | --- |
| [1] | Fuente1 |
| [2] | Fuente2 |

### Apéndices

A

### Apéndice A

Apéndice A