UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Algoritmo para la corrección de textos en español basados en los algoritmos Metaphone y Distancia de Levenshtein

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE:

Ingeniero de Sistemas

AUTORES:

Mego Lizana , Jhonn Anthony (ORCID: 0000-0003-0655-746X)

Cespedes Bravo, Segundo Manuel (ORCID: 0000-0001-9209-8727)

ASESORA:

Mgtr. Amorós Chávez, Gladys Jacqueline (ORCID: 0000-0003-3937-1119)

LÍNEA DE INVESTIGACIÓN:

Sistema de información y comunicaciones

LIMA – PERÚ

2021

Dedicatoria

A mis padres Mendino Aladino Mego Chancco y Lucila Lizana Santiago, quienes fueron las personas que más me apoyaron moral y económicamente en el transcurso de mi carrera. Gracias a ellos que con su amor, paciencia y esfuerzo me han permitido realizar hoy un sueño más a pesar de todas las adversidades. Gracias por inculcarme el ejemplo de esfuerzo y valentía. Atte. Jhonn Anthony Mego Lizana.

A mis padres Segundo Eladio Céspedes Santa Cruz y Barbara Bravo Huamán quienes me apoyaron moral e incondicionalmente, a mi enamorada Salesky Grimaneza Rivas Collazos y su mamá Elizabeth Maritza Collazos Cisneros por el apoyo incondicional y por estar en todo momento en el progreso de este gran logro. Gracias por motivarme siempre y enseñarme que no debo darme por vencido, perseguir mis sueños y alcanzar mis metas. Atte. Segundo Manuel Céspedes Bravo

Agradecimiento

Agradecemos a nuestros profesores de la Escuela de Ingeniería de Sistemas de la Universidad César Vallejo, por haber compartido sus conocimientos a lo largo de la preparación de nuestra profesión, de manera especial al Dr. Emigdio Antonio Alfaro Paredes quien nos guío con su paciencia, su rectitud y su valioso aporte de sabiduría en nuestra investigación.

Índice de contenidos

Índice de tablas

Índice de figuras

Resumen

El problema general de la investigación fue: ¿Cuál fue el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Distancia de Levenshtein en la corrección de errores ortográficos, gramaticales y de estilo? El objetivo general fue determinar el efecto del algoritmo MC Corrector de textos en español en la corrección de errores ortográficos, gramaticales y de estilo en base a los algoritmos Metaphone y Distancia de Levenshtein. La investigación tuvo un enfoque cuantitativo, un diseño experimental y un tipo de diseño pre-experimental.

El desarrollo del algoritmo MC Corrector para la corrección de textos en español con errores ortográficos, gramaticales y de estilo fue posible gracias a su unión con el algoritmo Metaphone y Distancia de Levenshtein, en el cual se extrajeron 100 párrafos del sitio web “enclavedeciencia” de la Real Academia Española para realizar las pruebas. El instrumento utilizado para esta investigación fue una hoja de recolección de datos, la herramienta es Excel y la técnica es la observación. Los resultados obtenidos del algoritmo MC Correcto en errores ortográficos corregidos es de un 93%, gramaticales de un 100%, y de estilo en un 100%.

Por tanto, se pudo determinar un efecto positivo del algoritmo MC en los errores ortográficos, gramaticales y de estilo corregidos. Así también cabe resaltar que se utilizaron una mayor cantidad de textos para precisar los resultados. Por consiguiente, se recomendó una mayor investigación e innovación en la corrección contextual y de muletillas, ya que no se encontró investigaciones profundas ni algoritmos para estos tipos de errores, así también se recomendó la realización de un algoritmo que genere palabras relacionadas en base al contexto, para el reforzamiento del algoritmo MC Corrector en la corrección contextual.

Palabras clave: Corrección de contexto automática, Sistema de corrección de texto, Detección y corrección de errores gramaticales, Sistema de corrección de contexto

Abstract

The general research problem was: What was the effect of a Spanish text correction algorithm based on the Metaphone and Levenshtein Distance algorithms in correcting spelling, grammar and style errors? The general objective was to determine the effect of the MC Corrector algorithm for Spanish texts in correcting spelling, grammar and style errors based on the Metaphone and Levenshtein Distance algorithms. The research had a quantitative approach, an experimental design and a type of pre-experimental design.

The development of the MC Corrector algorithm for correcting Spanish texts with spelling, grammatical and style errors was possible thanks to the union with the Metaphone and Levenshtein Distance algorithm, in which 100 paragraphs were extracted from the website "enclavedeciencia" the Royal Spanish Academy to carry out the tests. The instrument used for this research was a data collection sheet, the tool is Excel and the technique is observation. The results obtained from the MC Correct algorithm in corrected spelling errors is 93%, 100% grammatical, and 100% style.

Therefore, a positive effect of the MC algorithm on corrected spelling, grammar and style errors could be determined. It should also be noted that a greater number of texts were used to specify the results. Consequently, further research and innovation in contextual and filler correction was recommended, as no in-depth research or algorithms were found for these types of errors, as well as an algorithm that generates related words based on context was recommended for the reinforcement of the MC Corrector algorithm in contextual correction.

Keywords: Automatic context correction, Text Correction System, Detection and correction of grammatical errors, Context Correction System

I. INTRODUCCIÓN

En este trabajo de investigación se concretizan las ideas, esfuerzos y opiniones de aprendizaje para lograr desarrollar un algoritmo para la corrección de la escritura de textos en español basado en los algoritmos Metaphone y Levenshtein cuyo objetivo es lograr la corrección ortográfica, gramatical (contextual) y de estilo (muletillas) por párrafos en el idioma español.

Este primer capítulo desarrolla la realidad problemática que se presenta debido a que no se han encontrado sistemas y algoritmos que corrijan el contexto por párrafos en la lengua española, situación que ha limitado el avance progresivo en la corrección de textos por párrafo para una mejor ortografía y coherencia en el contexto académico al generarse producciones investigativas, en las cuales suelen apreciarse problemas de: coherencia, cohesión y redundancia en base a un sentido gramatical, así como errores ortográficos. Por consiguiente, se plantea como problema general la siguiente interrogante: ¿Cuál es el efecto de un algoritmo de corrección de textos en español basados en los algoritmos Metaphone y Distancia de Levenshtein en la corrección de errores ortográficos, gramaticales y de estilo? El problema se justifica en base a que, los algoritmos para corregir palabras u oraciones facilitan y ayudan a las personas a darse cuenta de cómo se debe escribir una palabra u oración, tal como indica San Mateo (2016) “podría ser útil que el programa proporcionara al usuario algún tipo de retroalimentación sobre el error cometido e incluso algunas opciones para corregirlo” (p. 115). Ello se comentará a través de un enfoque teórico, tecnológico y social.

El objetivo será desarrollar un algoritmo basado en Metaphone y Distancia de Levenshtein que mejore los resultados de corrección ortográfica, gramatical (contexto) y de estilo (muletillas) en lengua española tomando como variable la calidad de la corrección. Finalmente, se exponen las hipótesis que buscan validar una mejora de los resultados en función de la variable mencionada con los indicadores en cuestión.

Un problema real de las actividades humanas más comunes son los errores de ortográficos, gramaticales (contexto) y de estilo (muletillas). Estos errores pueden ser de muchos tipos, incluyendo su sintaxis, ya que, las personas suelen escribir sin darse cuenta de los errores de ortografía, redundancia y contextualización del párrafo. Muhammad, Tarek, Sadekur, Riazur y Farruk (2018): "El error ortográfico es un problema común en todos los idiomas, ya sea

en forma manuscrita o mecanografía" (p. 184). Además, explicaron que: "La revisión y corrección ortográfica siempre está en el foco de la lingüística computacional para casi todos los idiomas. Como resultado, se han observado esfuerzos significativos en esta área en varios idiomas como el inglés, el chino y el árabe" (Muhammad et al., 2018, p. 184). En base a ello, un campo importante de las nuevas tecnologías es la corrección automática de errores ortográficos basada en la técnica de procesamiento del lenguaje natural (inteligencia artificial) mediante la cual se pueden crear nuevos métodos para resolver estos problemas, sin embargo, estos avances aún no son suficientes por lo que no aplicará en nuestra investigación.

Es de apreciarse que uno de los problemas más comunes en las personas es la forma incorrecta de escribir las palabras y el contexto en la que se forma el párrafo, sin embargo, existen algoritmos que pueden corregir estos problemas, pero no se resuelven con total precisión, ya que estos algoritmos funcionan en base a la comparación de palabras, mas no en relación a su contexto, haciendo que sus resultados sean menos precisos y sin satisfacer al usuario final. Khatter (2019) explicó: “Los tipos más frecuentes de errores cometidos por las personas son: errores de puntuación que forman el área más problemática, seguidos de errores ortográficos, errores de preposición, errores de artículo, tiempo verbal incorrecto, forma de palabra incorrecta respectivamente” (pág. 364).

Una herramienta dedicada a la corrección del texto de un párrafo sería útil, no solo para personas analfabetas, sino también para estudiantes, ya que la corrección ortográfica ayuda a señalar errores de mecanografía y escritura. Muhammad, Tarek, Sadekur, Riazur y Farruk (2018) señalaron: "Esta técnica de corrección ortográfica automatizada también ayuda a los estudiantes importantes en el proceso de aprendizaje mediante la aplicación de palabras adecuadas durante el procesamiento de textos" (p. 184).

La corrección de texto desarrollaría una nueva forma de crear nuevos métodos de aprendizaje. Además, utilizando la comparación de palabras reales y no reales, se pueden aplicar técnicas de distancia y similitud. Lhoussain, Hicham y Abdellah (2015) indicaron:

> La corrección de errores ortográficos consiste en hacer comparaciones entre una palabra equivocada y las palabras de un léxico de una lengua determinada, y proponer una lista de palabras más cercanas a la palabra

> equivocada en función de la similitud y la distancia entre palabras. (pág. 127)

Se ha hecho un gran esfuerzo por la realización de programas, software, métodos, técnicas y algoritmos para desarrollar nuevas formas de corregir los errores de escritura y palabras, sin embargo, estos no están basados en el contexto, y si lo son, no son precisos ni confiables, no cumpliendo las necesidades del usuario. Lhoussain, Hicham y Abdellah (2015) también indicaron:

> El principal inconveniente de los sistemas de corrección ortográfica fuera de contexto es que estos sistemas devuelven múltiples soluciones en casos que tienen la misma importancia (incluso la distancia de Levenshtein, por ejemplo). Para remediar este problema, Gueddah y Yousfi introdujeron ponderaciones de los errores de las operaciones de edición para mejorar la programación de las soluciones devueltas por la distancia de Levenshtein. (pág. 128)

La realidad problemática de este proyecto es que aún no se ha encontrado suficiente investigación con respecto a los estudios sobre la corrección automática de la escritura de textos en párrafos, siendo esto muy importante para cualquier escritor o estudiante. Aloglah (2018) declaró: "Para ser más específicos, la ortografía se consideraría como un elemento muy importante en cualquier sistema de escritura. Cuando los escritores quieren expresar sus sentimientos y lo que quieren decir, usan la escritura" (p. 747).

Un estudio demuestra mediante un análisis los problemas comunes de escritura y redacción, y estos fueron los errores gramaticales, con el mayor porcentaje, errores de puntuación, seguidos de errores ortográficos. AlTameemy y Daradkeh (2019) señalaron:

> Después de analizar 80 párrafos a nivel de oración y a nivel de párrafo basados en una rúbrica en particular, los resultados revelan que los 80 estudiantes cometieron 1580 errores en total. A nivel de oración, los sujetos realizaron 1316 errores gramaticales (42,15%), puntuación (16,14%), ortografía (14,81%) y mayúsculas (10,19%). A nivel de párrafo, los participantes produjeron 264 errores ilustrados en el siguiente orden: errores en el desarrollo de párrafos (5.13%), errores en la coherencia de

los párrafos (4.87%), errores en la unidad de párrafos (3.80%) y errores en la inconsistencia del punto de vista (2.91%). (pág. 178)

Teniendo en cuenta que los errores gramaticales son los más frecuentes y cometidos por estudiantes y escritores, todavía no existe un algoritmo que pueda corregir errores en la escritura y la gramática, peor aún, no existe un algoritmo que corrija una palabra en función a su contexto en un nivel preciso.

Asimismo, es indispensable la justificación del estudio según el objetivo que se pretendió, por esta razón se menciona que los algoritmos para corregir palabras u oraciones han facilitado y ayudado a las personas a darse cuenta de cómo se debe escribir una palabra u oración. Para tal desarrollo, Moyotl-Hernández (2016) mencionó que: "el trabajo […] consistirá en elaborar un corpus de errores reales, ya que de él se podrán obtener los casos de error más frecuentes y en consecuencia un modelo de lenguaje que proporcionará información para estimar las probabilidades de las sugerencias" (pág. 154). De esta manera existen múltiples algoritmos que realizan esta tarea de forma independiente, por lo que se cree que la combinación de estos algoritmos y la contribución al desarrollo de uno nuevo ayudará de manera tecnológica, metodológica y social, ya que, esta herramienta desarrollada en base a ciertos algoritmos para la corrección de texto es de uso cotidiano, pudiendo así implementarse en organizaciones que dependen de una buena escritura sin errores ortográficos y con buena gramática.

Las justificaciones teóricas señalan que el objetivo es proporcionar un algoritmo que corrija la escritura de textos en lengua española. Posteriormente se avanzaría con algo más complejo, por ejemplo, oraciones o el texto completo en un párrafo. En esa línea, Moyotl-Hernández (2016) explicó que la finalidad es: "mejorar la precisión al elegir la palabra correcta. De igual manera, las cuestiones relacionadas con los errores ortográficos múltiples, gramaticales y de estilo, se tratarán en trabajos posteriores" (p. 155), lo cual permite entender la lógica de cómo realizar el algoritmo propuesto para que tengan un mejor nivel de escritura basado en la coherencia y la cohesión y por lo tanto evitar la redundancia.

En cuanto a la justificación tecnológica al implementar un nuevo algoritmo basado en otros algoritmos, habrá una versión fusionada y mejorada con los métodos mencionados, así pues, en base al notorio crecimiento tecnológico de los sistemas y la tecnología, deben conocerse las técnicas de algoritmo para la

corrección de texto, resolver estos problemas y al mismo tiempo encontrar nuevas técnicas. Drias, Kechid y Pasi (2016) explicaron: "El gran progreso experimentado por la corrección automática recientemente dio lugar a nuevos desafíos y paradigmas. Una de esas preocupaciones es tener un sistema inteligente capaz de corregir la escritura de un párrafo" (p. 1).

De acuerdo con lo anterior, se entiende que el crecimiento tecnológico de los sistemas de corrección va a pasos agigantados, especialmente cuando la inteligencia artificial ha ido evolucionando con las nuevas técnicas que aparecen y el procesamiento del lenguaje, que se están implementando en el mundo de la tecnología. Se puede concluir que al crear un nuevo algoritmo basado en dos algoritmos que son los más utilizados por las grandes empresas, se puede tener una versión mejorada de los resultados de corrección. Al respecto, Arrojo (2017) indicó que:

> La inteligencia artificial ha estado en la raíz del cambio revolucionario en la tecnología, por lo que también se ha traducido en gramática, en la medida en que la IA es la clave de la 'revolución digital' que ha contribuido a la educación y la búsqueda de errores en palabras y texto. (p. 426)

Por otro lado, la justificación social se centra en proporcionar un corrector de párrafos que beneficie a la sociedad. Al respecto, Zelasco, Hohendahl y Donayo (2015) explicaron:

> Las personas siempre y de forma natural han ejercido la capacidad de leer correctamente textos en mal estado, escuchar bien palabras erróneas, faltantes e incluso mezcladas con ruidos de todo tipo. Si vamos en el futuro a interactuar de manera más inteligente con las máquinas, es hora de que aprendan las cosas que las personas hacen naturalmente: corregir errores. (pág. 11)

La gramática, la escritura y la ortografía juegan un papel muy importante en la comunicación entre dos o más personas. La tecnología ya está en todas partes, haciendo que las personas sean accesibles a todo tipo de información y herramientas para su autoaprendizaje y entretenimiento por lo que es necesaria una herramienta que les permita alcanzar un aprendizaje ortográfico y gramatical que ayude a contener menos errores en sus textos. Además, como menciona San Mateo (2016) es importante que:

> El usuario no sea un mero receptor pasivo de la información, sino que sea capaz de intuir la causa de los avisos que envía. Además, es conveniente que el usuario sea consciente de la necesidad de complementar el uso del corrector con el Diccionario de la lengua española y el Diccionario Panhispánico de dudas de la RAE. (p. 115)

Sobre la base de realidad problemática presentada se planteó el problema general y los problemas específicos de la investigación. El problema general de la investigación fue ¿Cuál fue el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Distancia de Levenshtein en la corrección de errores ortográficos, gramaticales y de estilo? Los problemas específicos de la investigación fueron los siguientes:

- **PE1:** ¿Cuál fue el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Distancia de Levenshtein en la corrección de errores ortográficos?
- **PE2:** ¿Cuál fue el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Distancia de Levenshtein en la corrección de errores gramaticales?
- **PE3:** ¿Cuál fue el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Distancia de Levenshtein en la corrección de errores de estilo?

El objetivo general fue determinar el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Distancia de Levenshtein en la corrección de errores ortográficos, gramaticales y de estilo. Los objetivos específicos fueron los siguientes:

- **OE1:** Determinar el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Distancia de Levenshtein en la corrección de errores ortográficos.
- **OE2:** Determinar el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Distancia de Levenshtein en la corrección de errores gramaticales.

- **OE3:** Determinar el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Distancia de Levenshtein en la corrección de errores de estilo.

La hipótesis general fue: “El algoritmo de corrección de la escritura de textos en español basado en los algoritmos de Metaphone y Distancia de Levenshtein (algoritmo MC Corrector) incrementó el porcentaje de errores ortográficos, gramaticales y de estilo corregidos”. En respecto, Muhammad et al. (2018) señalan que en base a algunos reportes de revisión ortográficos se determinó que “aunque casi todos ellos se han concentrado en la revisión ortográfica sin contexto, muy pocos de ellos se centraron en la revisión ortográfica dependiente del contexto, donde se encuentra mucho potencial de rayo de éxito” (p. 184). Complementando a ello, Moukrim, Tragha, Benlahmer y Almalki (2019) mencionaron que:

> El defecto de dicho programa de software radica en la relación entre las palabras que constituyen una oración que a veces puede ser sintácticamente incorrecta y que, por lo tanto, puede conducir a resultados incorrectos. Esto requiere imperativamente un sistema de corrección automática efectiva. (p. 1)

De la misma manera las hipótesis específicas de la investigación fueron:

- **HE1:** El algoritmo de corrección de la escritura de textos en español basado en los algoritmos de Metaphone y Distancia de Levenshtein (algoritmo MC Corrector) incrementó el porcentaje de errores ortográficos corregidos por lo menos en un 73%

El algoritmo MC Corrector que se desarrolló en la investigación basados en los algoritmos Metaphone y Distancia de Levenshtein aumentaría significativamente el porcentaje de la cantidad de errores ortográficos corregidos. Yulianto, Arifudin y Alamsyah (2018) mencionaron:

> Al usar este algoritmo, el rendimiento de búsqueda es más preciso. Incluso la palabra ingresada es incorrecta, este algoritmo aún puede encontrar los datos que el usuario necesita y proporcionar sugerencias de búsqueda que se acercan a partir de la palabra ingresada. (p. 68)

- **HE2:** El algoritmo de corrección de la escritura de textos en español basado en los algoritmos de Metaphone y Distancia de Levenshtein (algoritmo MC Corrector) incrementó el porcentaje de errores gramaticales corregidos por lo menos en un 100%.

Lhoussain, Hicham y Abdellah (2015) indicaron que:

> El principal inconveniente de los sistemas de corrección ortográfica fuera de contexto es que estos sistemas devuelven múltiples soluciones en casos que tienen la misma importancia (incluso la distancia de Levenshtein, por ejemplo). Para remediar este problema, Gueddah y Yousfi introdujeron ponderaciones de los errores de las operaciones de edición para mejorar la programación de las soluciones devueltas por la distancia de Levenshtein. (pág. 128)

- **HE3:** El algoritmo de corrección de la escritura de textos en español basado en los algoritmos de Metaphone y Distancia de Levenshtein (algoritmo MC Corrector) incrementó el porcentaje de errores de estilo corregidos por lo menos en un 92%.

II. MARCO TEÓRICO

En este capítulo se definen los trabajos previos donde se desarrollaron algoritmos para la corrección de palabras y textos, en los cuales se harán énfasis a los objetivos, metodología, técnicas y resultados obtenidos que nos ayudarán a la elaboración de nuestra investigación. En el capítulo también se definen las teorías relacionadas donde se conceptualizan los algoritmos: Metaphone y Distancia de Leveshtein así como la elaboración de un marco conceptual donde se delimitan las reglas gramaticales y dimensiones de la presente investigación.

Uno de los métodos más pronunciados que la propia empresa Microsoft utiliza en sus correctores gramaticales, tanto por palabra como por oraciones, es el método Levenshtein. En ese sentido, Christanti, Rudy y Naga (2018) realizaron una investigación cuyo objetivo era crear un sistema de corrección ortográfica rápido y preciso con la capacidad de manejar tanto el tipo de errores ortográficos, no palabras y errores de palabras reales. El diseño utilizado en la investigación fue experimental. Donde se analizó el sistema existente para modificar su precisión y velocidad. Se concluyó en la investigación que el mejor resultado en términos de precisión y velocidad es un sistema que utiliza bigram con distancia Trie y Damearu-Levenshtein. (p. 832)

Moyotl-Hernandez (2016) realizó la investigación teniendo por objetivo mejorar el funcionamiento en la distancia de Levenshtein asignando costos diferentes en las operaciones de edición, tomando en cuenta la frecuencia de letras para mejorar la corrección ortográfica. El diseño empleado fue experimental, cuya muestra fue de 55 oraciones con 45 errores distintos. El instrumento utilizado fue un cuadro comparativo de los resultados de la cantidad de errores corregidos entre el método base (Distancia de Levenshtein), el método propuesto (alteración de costos de la Distancia de Levenshtein) y las herramientas Word 2016 y Google docs., donde se obtuvo como resultados el 73.3%, 71.1%, 66.6% y el 40% respectivamente. En esta investigación se concluyó que el corrector ortográfico depende del dominio de aplicación y el idioma a tratar, y pese a tratarse de un corrector ortográfico de propósito general, los resultados obtenidos fueron prometedores.

Los formatos para los modos de aprendizaje también forman parte de los sistemas que identifican y corrigen la escritura en función del idioma, la única diferencia, es que, en estos casos en lugar de corregir automáticamente la palabra equivocada, se hace notar al usuario el error cometido y a su vez

proporciona sugerencias para corregir el texto. Al respecto, Wang y Xu (2018) realizó una investigación donde "propone un sistema de apoyo inteligente para la escritura en inglés basado en el modo B/S" (p. 157), "con el objetivo de explorar un nuevo método de entrenamiento eficiente en escritura en inglés basado en B/S" (Wang y Xu, 2018, p. 158). El diseño utilizado en la investigación fue experimental y tuvo un enfoque educativo, tanto para estudiantes como para profesores. En esta investigación, Wang y Xu (2018) concluyeron que combinado con Entity Framework, el sistema inteligente de capacitación en escritura en inglés puede combinar de manera efectiva las habilidades de escritura en inglés, la capacitación en escritura y los materiales de escritura, y realizar la función de corrección automática en la corrección de errores. (p. 168)

Una forma más eficiente de resolver los problemas gramaticales en un aspecto algorítmico se da mediante la comparación de diferentes tecnologías que buscan generar una mejor solución para las palabras reales, ya existentes en la base de datos y expandirlo a soluciones gramaticales en otros idiomas. A partir de ello, Dashti (2017) realizó una investigación cuyo propósito fue proponer un nuevo modelo que se enfoca en detectar y corregir palabras reales en una oración. Manipulando una gramática probabilística. El diseño de la investigación fue experimental. Además, se realizó una comparación con otros modelos, como Hirst y Budanitsky WordNet y el método de tamaño de ventana fija Wilcox-O'Hearn, Hirst y Budanitsky. Queriendo demostrar que su método es superior a todos los mencionados anteriormente. En la investigación realizada se concluyó que el método propuesto mostró mayor precisión en la corrección de múltiples errores en una ventana, sin embargo, el tiempo de respuesta no fue el esperado. (p. 501)

Los algoritmos de corrección gramatical son una alternativa de solución no solo para los hablantes nativos del idioma español, sino también para los que dominan el idioma extranjero y que quiera aprender el idioma español. En base a ello, Ferreira y Hernández (2017) realizaron una investigación que tuvo como objetivo contribuir al diseño e implementación de un corrector ortográfico para el idioma español que detecte errores de manera inteligente, automatice el proceso y sea eficiente, con el fin de integrarlo en el Sistema Smart Tutorial, ELE-TUTORA. El diseño utilizado fue descriptivo cuya muestra ha sido de 62 estudiantes extranjeros. Además, el algoritmo pronunciado fue codificado bajo la

herramienta Python y NLTK. En esta investigación se concluyó que el corrector construido se considera una herramienta que puede ayudar a mejorar la precisión lingüística en la producción escrita de un texto, por dos razones principales: una técnica y otra lingüística. (p. 404)

Diversos sistemas de corrección gramatical interactúan con una base de datos, sin embargo, al intentar solucionar un error común basado en la falta de un registro previo en una base de datos, permite de errores o no los corrige adecuadamente, por lo que, se empezó a utilizar algo que interactuara de forma independiente. De esta manera, Dronen (2016) realizó un estudio cuyo objetivo se basa en la detección y corrección de palabras utilizando redes neuronales convolucionales (ConvNets). El diseño utilizado fue descriptivo comparativo, donde se prueba la funcionalidad de corrección de palabras con modelos de lenguaje probabilístico. Con las pruebas y comparaciones realizadas en esta investigación, se concluyó que ConVets bajo cualquier entorno funciona mucho mejor para la correcta detección y corrección de palabras a diferencia de los modelos de lenguaje probabilístico que por lejos conllevan una gran desventaja. (p. 109)

Alguno de los métodos más utilizados en sistemas de corrección gramatical o de ortografía son una base de datos para comparar una palabra mal escrita, así como el análisis de probabilidades estadísticas para sugerir una palabra o corregir errores dependiendo del texto que se digita en tiempo real. En base a ello, Faili, Ehsan, Montazery y Pilehvar (2016) realizaron una investigación que tuvo como objetivo el desarrollo de un corrector automático de ortografía, gramática y palabras reales para el idioma persa, dicho corrector llamado Vafa Spell-Checker. El diseño utilizado en la investigación fue experimental. Al mismo tiempo, explica que dicho corrector híbrido se basa en enfoques estadísticos. Se concluyó en la investigación que el sistema propuesto es una herramienta eficaz que no solo detecta palabras y algunas oraciones no gramaticales, sino que también repara los errores ortográficos. Además, todo el sistema se puede utilizar como un módulo adicional para el editor de texto de Microsoft Word. (p. 114)

En la misma línea, San Mateo (2016) realizó una investigación cuyo objetivo es describir un algoritmo de corrección ortográfica y gramatical para textos en lengua española, para personas competentes de esta lengua. El diseño

utilizado en la investigación fue experimental, teniendo una muestra total de 12 entre oraciones y frases. El instrumento utilizado fue un cuadro comparativo de la cantidad de errores corregidos entre las herramientas Microsoft Word, SpanishChecker, Stilus y CorrectMe, obteniendo como resultados el 25%, 37%, 23% y 100% respectivamente. El algoritmo se basa en el reconocimiento de errores a través del análisis estadístico bajo la herramienta CorrectMe. Además, se concluyó en la investigación que a través del análisis estadístico fue posible detectar errores que otros algoritmos presentados no detectan. El punto débil es que para identificar los errores deben ser inferidos y analizados anteriormente, lo que a menudo presenta falsos positivos o falsos negativos.

Estos sistemas de corrección gramatical mencionados también han sido trabajados en otras lenguas como el inglés, en base a ello, Lawley (2015) realizó una investigación, que tuvo como objetivo crear un software basado en la web llamada Grammar Checker para la corrección de palabras en el idioma inglés que ayuda a los estudiantes de una universidad en España. El diseño utilizado en la investigación fue experimental, ya que se realizaron muchas modificaciones para lograr una interfaz didáctica para que pueda interactuar de una manera sencilla y divertida con el fin de ayudar a los estudiantes españoles con la escritura en el idioma inglés. La investigación concluyó que el software creado a pesar de no ser perfecto y ser solo un prototipo si ayudaba a los estudiantes en su redacción para los ensayos, ya que esto no corrige automáticamente los errores, sino que de manera didáctica sugiere que la preposición escrita es rara, de esta manera, el usuario modifica el texto, dándose cuenta de sus errores y aprendiendo. (p. 23)

De la misma forma, Mosavi (2014) establece un precedente en el idioma persa realizando una investigación que tenía el propósito de "demostrar la efectividad de un gran cuerpo persa monolingüe para mejorar la calidad de salida de un corrector ortográfico desarrollado para este idioma" (p. 56). El diseño utilizado en la investigación fue experimental, donde se evaluó el desempeño del Sistema de Corrección Ortográfica. En la investigación Mosavi (2014) explica que: "FarsiSpell, [es un] corrector ortográfico propuesto en la investigación [que] tiene una ventaja de adaptabilidad, ya que puede funcionar en cualquier entorno a diferencia de otros sistemas, como Virastyar o Vafa que solo funcionan bajo el entorno Microsoft Word" (p. 72).

Otro punto importante a detallar, es la corrección de estilo, que permite un mejor entendimiento y omite la redundancia de palabras al momento de redactar. Fernández (2007), realizó una investigación que tuvo por finalidad facilitar el camino a los aprendices polacos que se enfrentan al español y a las culturas con él relacionadas. El diseño utilizado fue descriptivo, se tuvo una población de 67 textos y una muestra 334 oraciones. El instrumento utilizado fue un cuadro comparativo para los conectores más usados o repetitivos por aprendices polacos de español en los 3 diferentes grados, donde se observó que de 334 oraciones se encontraron un total de 51 errores de conectores, los cuales fueron, 18 consecutivos, 8 comparativos, 7 condicionales, 7 temporales, 7 causales, 3 finales y 1 concesivo. Finalmente, se concluyó que las comparativas y condicionales fluctúan entre el 25 y el 30 por ciento, las temporales y causales entre el 7 y el 12 y, finalmente, las concesivas y finales que parecen no plantear problemas.

Para concluir con este punto, la elaboración de algoritmos y los sistemas presentados tienden a ser elaborados a partir de bigramas, siendo muy pocas las ocasiones en las que se utiliza la corrección de palabras en trigramas, por lo que Athanaselis, Bakamidis y Dologlou (2006) realizaron un estudio donde "presentó un enfoque para reparar errores de orden de palabras en el texto reorganizando las palabras en una oración y eligiendo la versión que maximiza el número de aciertos de trigrama de acuerdo con un modelo de lenguaje" (p. 1). El diseño utilizado fue cuantitativo y descriptivo, donde se extrae información del método utilizado para la corrección gramatical del idioma inglés y posteriormente se describen los beneficios otorgados por el uso de este método en comparación con otros. Asimismo, la investigación concluyó que el método propuesto es efectivo para reparar oraciones erróneas de modo que los resultados muestran que la mayoría de las oraciones pueden ser reparadas por este método independientemente de la longitud de la oración y el tipo de errores de orden de las palabras de las oraciones. (Athanaselis, Bakamidis y Dologlou, 2006)

En cuanto a las teorías relacionadas, uno de los algoritmos es "Metaphone, un algoritmo fonético basado en reglas. Se utiliza como el principal método de evaluación fonética. Se elige en función de la evaluación de varias opciones de código abierto debido a su flexibilidad general y niveles de rendimiento" (Chan, Vasardani y Winter, 2015, p. 540).

El algoritmo Metaphone resuelve muchos problemas que otros algoritmos no tenían, esto es gestionar los resultados en función de su contextualización, haciendo que sus predicciones sean más coherentes y significativas. Chan, Vasardani y Winter (2015) indicaron: "El sistema Metaphone mostró la capacidad de manejar la contextualización convencional mejor que Soundex (Lait y Randell 1996; Snae 2007) y relativamente a la par con las otras alternativas disponibles. Además, no es rígido en las transformaciones de personajes basadas en reglas" (p. 541).

Este algoritmo devuelve sugerencias de palabras basadas en los errores de una palabra encontrada. Metaphone 3 mejoró la precisión de la sugerencia en un 76.2%, pero este algoritmo solo se basa en corregir palabras. Lai, Topaz, Goss y Zhou (2015) explicaron:

> El algoritmo Metaphone asigna la falta de ortografía a un código; las palabras con el mismo código o similares se devuelven como sugerencias. Crowell et al. encontraron que el rendimiento mejoró enormemente al reordenar la lista de sugerencias en función de las frecuencias de las posibles correcciones, hasta una tasa de precisión de sugerencia superior del 76,2%. (p. 189)

Ya existen varios algoritmos que funcionan con la comparación fonética. Metaphone fue elegido por tener una precisión mucho mayor y estar constantemente actualizado. Giraud, Goodliffe y Cueva (2015) mencionaron: "Los algoritmos comunes de comparación fonética incluyen Soundex, Phonex, Phonix y Double-metaphone" (p. 393).

El siguiente algoritmo utilizado es la distancia de Levenshtein quien obtiene la longitud de distancia entre dos palabras que es utilizada para definir una serie de operaciones de edición las cuales son necesarias para transformarse en una palabra ya registrada en un diccionario. Sobre ello, Lhoussain, Hicham y Abdellah (2015) mencionan que:

> El método Metric introducido por Levenshtein mide la similitud entre dos palabras calculando una distancia de edición. La distancia de edición se define como el número mínimo de operaciones básicas de edición necesarias para transformar una palabra incorrecta en una palabra de diccionario. Por lo tanto, para corregir una palabra incorrecta, se conserva

> un conjunto de soluciones que requieren menos operaciones de edición posibles. (p. 129)

Este algoritmo tiene la ventaja de ser aún más valioso a la hora de encontrar errores ortográficos en comparación con otros, sin embargo, no se basa en la contextualización, sino en la distancia entre palabras, pese a ello, es uno de los mejores algoritmos desarrollados para la corrección automática de errores ortográficos. Yulianto, Arifudin y Alamsyah (2018) indicaron que:

> La distancia de Leveshtein, un caso especial de distancia de edición donde se aplican costos unitarios. Mediante el uso de este algoritmo, el rendimiento de la búsqueda es más preciso. Incluso la palabra introducida es incorrecta, este algoritmo todavía puede encontrar los datos que el usuario necesita y proporcionar sugerencias de búsqueda que se acercan a partir de la palabra ingresada. (p. 68)

Es más, Lhoussain et al. (2015) señalaron que: "la distancia métrica devuelta por el algoritmo de Levenshtein es a menudo la misma para múltiples soluciones en la corrección de una palabra incorrecta" (p. 127). Así deducimos que este algoritmo será aplicable para este tipo de proyecto de investigación, debido al complejo proceso que requiere cada palabra y su resultado preciso para poder presentar sugerencias al usuario final.

Las principales funciones que cumple este algoritmo son la de inserción, eliminación y permutación dando resultados variados basados en la distancia de una palabra y su predecesora de lo cual se obtiene un cálculo del que se pueden extraer las sugerencias mal escritas de posibles correcciones de esa palabra. En ese sentido, Lhoussain et al. (2015) indicaron:

> Basándose en los trabajos de Damerau, Levenshtein consideró tres operaciones de edición (inserción, eliminación, permutación), y definió su método como la distancia de edición que compara dos palabras mientras calcula el número de operaciones de edición sometidas en una palabra para convertirla en otra. Esta distancia también se llama distancia de Levenshtein. (p. 128)

El algoritmo maneja un procedimiento complejo para calcular sus resultados, sin embargo, esos resultados son más precisos que otros algoritmos de revisión ortográfica. Su proceso se basa principalmente en obtener los nodos

más cercanos que interconectan cada palabra, obteniendo así su resultado más preciso. Ho, Oh y Kim (2017) explicaron que:

> Los algoritmos secuenciales tradicionales para ASM con “n” diferencias adoptan el modelo de programación dinámica. Estos algoritmos pueden calcular la matriz de distancia de edición entre la cadena de entrada y el patrón para obtener las operaciones de edición mínimas para convertir cada factor de la cadena de entrada en patrón. Desafortunadamente, cada elemento de la matriz depende de los elementos anteriores en la misma fila o columna. (p. 2)

Los resultados finales del algoritmo de distancia de edición de Levenshtein, es el más preciso en relación con la corrección automática de errores ortográficos, haciendo coincidir las palabras en la cadena para mostrar sugerencias de texto correctivo al usuario final. En pocas palabras, Yulianto et al. (2018) señalaron que:

> El algoritmo de distancia de Levenshtein da buenos resultados en la resolución de problemas en la coincidencia de los datos de la cadena para proporcionar sugerencias de texto, por ejemplo, en el reconocimiento de escritura a mano, palabras de búsqueda y palabras mal escritas para que la efectividad de la entrada aumente, se puedan evitar errores ortográficos y el autocompletado acelere las interacciones entre humanos y computadoras. (p. 68)

Por otro lado, tenemos que las reglas gramaticales sirven como punto de partida para que el algoritmo a realizarse posea un sentido de coherencia y cohesión al momento de corregir, evitando las redundancias. Acorde con ello, la coherencia según Iglesias, Gonzales y Hernández (2019) es definida como:

> Una secuencia lógica de proposiciones, expresadas en oraciones, que se enlazan entre sí por medio de elementos sintácticos, lo que la hace perfectamente comprensible. [… Esta es una] característica de los textos que nos hace percibir todas sus partes como compatibles en un mismo todo y que nos permite avanzar sin que la interpretación de unas partes deba hacerse a costa de olvidar otras. La incompatibilidad que podría romper la coherencia de un texto es la relación entre dos frases que no pueden ser simultáneamente verdaderas, pero la aceptación de una implica la negación de la otra y viceversa. (p. 297, 301)

En contraste, se presenta la incoherencia que es definida como:

> El más grave de los errores de construcción textual porque amenaza directamente la posibilidad de que el texto sea interpretable. La incoherencia es a veces el resultado de un conocimiento insuficiente del léxico utilizado, lo que hace que el editor no diga lo que cree que está diciendo. (Iglesias et al., 2019, p. 301)

Desde otro punto de vista la cohesión “es el conjunto de mecanismos que un texto para asegurar la conexión entre sus componentes, es decir, la relación que deben presentar las diferentes partes que componen el texto” (Evangelista, 2014, p. 228), lo cual se entiende como el nexo que existe después de un punto dentro de un párrafo, para que tenga consistencia y así evitar redundancia dentro de un párrafo.

Con respecto a las dimensiones, se detallan las siguientes:

Calidad de corrección ortográfica:

Lo que esta dimensión quiere tratar es la calidad con la que nuestro algoritmo identifica las palabras que están mal escritas y sugiere al usuario soluciones alternativas o corregirlo automáticamente teniendo coherencia con el texto que está escribiendo como indica Moyotl-Hernández (2016) señaló:

> Su función es identificar palabras que posiblemente estén mal escritas y presentar una serie de alternativas de corrección al usuario. Para saber si una palabra está escrita correctamente o no, el enfoque más simple es usar un diccionario como referencia con la lista de (casi) todas las palabras válidas en el idioma tratado. Una palabra se escribe correctamente si se encuentra en el diccionario, de lo contrario se considera un error. (p. 146)

Calidad de la corrección gramatical:

Para esta dimensión, la esencia del contexto en los párrafos es de suma importancia ya que, en base a eso, nuestro algoritmo se basa en mostrar una alta tasa de calidad hacia los usuarios, mostrando coherencia en la conexión que deben tener las oraciones con un "punto" de separación, por lo tanto, la corrección gramatical, respecto a Moyotl-Hernández (2016) explicó:

> Se encarga de verificar la correcta construcción de una oración, como la concordancia de género y número, los tiempos verbales, etc.; por lo tanto, no siempre pueden descubrir errores en los que la palabra está ortográfica correctamente pero su uso es incorrecto en un contexto específico. (p. 147)

Calidad de corrección de estilo:

Esta dimensión, busca omitir o corregir las repeticiones de palabras seguidas o conectores, pues sabemos que las muletillas son errores muy comunes y que nuestro algoritmo propuesto busca corregir. Así pues,

> Cuando repetimos una palabra en el mismo párrafo, empobrecemos el estilo del texto. Podemos reorganizar las frases para eliminar la palabra o sustituirla por medio de pronombres o sinónimos. Hemos de tener cuidado con los sinónimos, porque dos palabras afines no siempre encierran el mismo significado. (Ivap 2019)

III. METODOLOGÍA

3.1 Tipo y diseño de investigación

Esta investigación cumple un propósito aplicado ya que:

> Se entiende como aquella actividad científica orientada hacia un fin práctico más o menos inmediato. Su finalidad radica en la aplicación concreta de un saber que no busca tanto incrementar su corpus teórico como ensayar sus posibilidades prácticas en el plano de la acción. Su definición está, pues, en relación a criterios precisos de uso, tendentes a facilitar respuestas a problemas prácticos específicos, constituyéndose en un área intermedia entre el descubrimiento de un nuevo conocimiento y su aplicación práctica a través de la cual se trata de transformar los conocimientos científicos en tecnologías. (Rodríguez, 2011, p. 37)

Así también, por la naturaleza de los datos y la información recolectada, esta investigación tiene un carácter cuantitativo, la cual:

> Parte de cuerpos teóricos aceptados por la comunidad científica con base en los cuales fórmula hipótesis sobre relaciones esperadas entre las variables que hacen parte del problema que se estudia. Su constatación se realiza mediante la recolección de información cuantitativa orientada por conceptos empíricos medibles, derivados de los conceptos teóricos con los que se construyen las hipótesis conceptuales. El análisis de la información recolectada tiene por fin determinar el frado de significación de las relaciones previstas entre las variables. El procedimiento que se sigue es hipotético-deductivo el cual inicia con la formulación de hipótesis derivadas de la teoría, continúa con la operacionalización de las variables, la recolección, el procesamiento de los datos y la interpretación. Los datos empíricos constituyen la base para a prueba de las hipótesis y los modelos teóricos formulados por el investigador. (Monje, 2011, p. 13)

El diseño de investigación es pre-experimental donde después de haber diseñado el algoritmo, se realizará un postest con un solo grupo, nuestro algoritmo, donde se someterá a corregir textos y ver la mejora esperada en el algoritmo desarrollado en esta investigación. Alonso, *et al.* (2011) señalan:

> El diseño de sólo posttest con un grupo, [refiere que] el investigador proporciona un tratamiento y a continuación hace una observación […] y no se considera la comparación con otros tratamientos. [Además,] solo podemos hacer aproximaciones por lo que respecta a las relaciones causales. (p. 21)

Así también, Ñaupas, Mejía, Novoa y Villagómez (2014) indican que,

- G: grupo o muestra
- X: tratamiento de la variable experimental
- O: medición de la variable dependiente (p. 337)

3.2 Variables y operacionalización

La variable la calidad de corrección, la cual resulta importante ya que es necesaria para evaluar la calidad de corrección que realizará el algoritmo a desarrollarse durante el proceso de evaluación del texto, para lo que se debe especificar en qué base es que el texto será evaluado y corregido. La Asociación española de Editores (2018) explicó que esta variable "tiene como objetivo garantizar la calidad del lenguaje y la eficacia de la comunicación escrita" (p. 4).

En el **Anexo 3** se detalla cada punto con respecto a la definición conceptual y operativa de la variable, sus respectivas dimensiones y los indicadores.

3.3 Población, muestra y muestreo

Para el presente trabajo, la población será determinada por el número de párrafos, para ello se ha tomado diversos corpus, teniendo un total de 100 párrafos. Al respecto, Badii, Castillo y Landeros (2007) mencionaron que:

> La población es cualquier grupo completo, ya sea de personas, animales o cosas. Es la totalidad de los elementos o cosas bajo consideración. La población se refiere a un grupo finito de elementos. Elementos de una población: son las unidades individuales que constituyen o conforman una población. (p. 119)

Por consiguiente, para esta investigación, en base a la población, se ha tomado una muestra no probabilística el total de la población, siendo nuestra muestra 100 párrafos. Así pues, "la muestra es un subconjunto de la población a la que se accede y debe ser representativa de ella porque se realizan las mediciones pertinentes sobre ella" (Gamboa, 2018, p. 6).

Así también, para esta investigación se aplicará un muestro no probabilístico, “también es conocido como muestreo por conveniencia, no es aleatorio, razón por la que se desconoce la probabilidad de selección de cada unidad o elemento de la población”. (Pineda, De Alvarado y De Canales, 1994, p. 119)

3.4 Técnicas e instrumentos de recolección de datos

Para la investigación realizada se ha utilizado como técnica la observación. Al respecto, Morales (2014) menciona que es el uso de técnicas e instrumentos para recopilar información sobre un determinado tema lo que es objeto de investigación. Es una de las tareas más importantes en la etapa de análisis de sistemas de información porque del producto a desarrollar depende de ello. Dado que nuestro trabajo es pre-experimental, se ha utilizado el instrumento de la hoja de recolección de datos, que ayudará a estimar la precisión y la calidad de corrección de textos.

La validez se puede definir como un instrumento utilizado correctamente para cada situación. En otras palabras, con respecto a Galbiati (2015) menciona que, un instrumento de medida es válido con respecto a una propiedad, si es relevante para ella, es decir, un cambio en la propiedad determina un cambio en la medida, y un cambio en la medida se debe solo a un cambio en la propiedad. Un instrumento válido proporciona una medida válida (pág. 3). De esta forma es necesario que la validez del contenido sean datos generados a través de una aplicación, y a su vez aplicando todas las reglas gramaticales según la RAE.

La confiabilidad se basa en el hecho de que los resultados repetidos de un instrumento no se apartan mucho, es decir, con respecto a Santos (2017), mencionó que, "La confiabilidad consiste en determinar hasta qué punto las respuestas de un instrumento de medición aplicado a un conjunto de individuos son estables independientemente del individuo que lo aplique y el tiempo en que

se aplique" (p. 3). Sin embargo, la confiabilidad de nuestro trabajo se basa en los instrumentos utilizados por otros autores, para ejecutar el algoritmo y garantizar la exactitud de los textos, utilizando un nivel de confianza del 95%.

3.5 Procedimientos

Para la presente investigación de naturaleza cuantitativa se utilizó el método de análisis de datos que consistió en la recolección de 100 párrafos aleatorios de diferentes corpus almacenados en la RAE, puesto que son libres, se usaron para realizar la hipótesis basada en la medición del aumento del porcentaje estadístico. Se utilizaron y combinaron los algoritmos de Metaphone y Levenshtein para el desarrollo de un nuevo método de corrección de textos.

Paso 1: Identificar el problema de corrección ortográfica en un enfoque contextual, de estilo y gramatical.

Paso 2: Crear identificadores de errores.

Se listan los errores ortográficos con sus respectivos tipos de errores a corregir, donde se muestra el concepto de los errores principales a corregir, ejemplos de errores, la corrección y el tipo de error corregido. *Ver tabla 1*

Tabla 1 *Tipos de errores a corregir*

Concepto	Errores	Corregidos	Tipo de errores
Escribir mal una palabra	umanoz, politic4	Humanos, política	ortográfico
Escribir una palabra correcta, pero fuera de contexto,	salgamos a votar la basura	Salgamos a botar la basura	Contextual
Escribir repetitivamente conectores al momento de redactar	Como hoy es domingo saldré con mis amigos, ya que no he salido en estos días, ya que he estado muy elaborando mi tesis	Como hoy es domingo saldré con mis amigos, ya que no he salido en estos días, puesto que he estado muy	De estilo

		ocupado elaborando mi tesis	
Empezar un párrafo con minúscula o terminarla sin un punto final, dejar espacios innecesarios, entre otros	algunos animales feroces son: los tigres, cocodrilos o osos	Algunos animales feroces son: los tigres, cocodrilos u osos.	Gramatical

Paso 3: Ingresar el texto a corregir

Los párrafos para corregir son ingresados a la herramienta de corrección de datos, el algoritmo procesará la información ingresada y corrige la ortografía, el contexto de las palabras, la repetición de conectores que perjudican el estilo de redacción y la gramática. Finalmente muestra el párrafo corregido en su totalidad y como adicional, muestra los errores detectados e indica los cambios realizados.

El resultado de esta prueba se llenará en una tabla de recolección de datos de conteo para posteriormente ser comparada con otras herramientas de corrección y determinar el incremento del porcentaje de errores corregidos.

Paso 4: Consideraciones para llenar la tabla

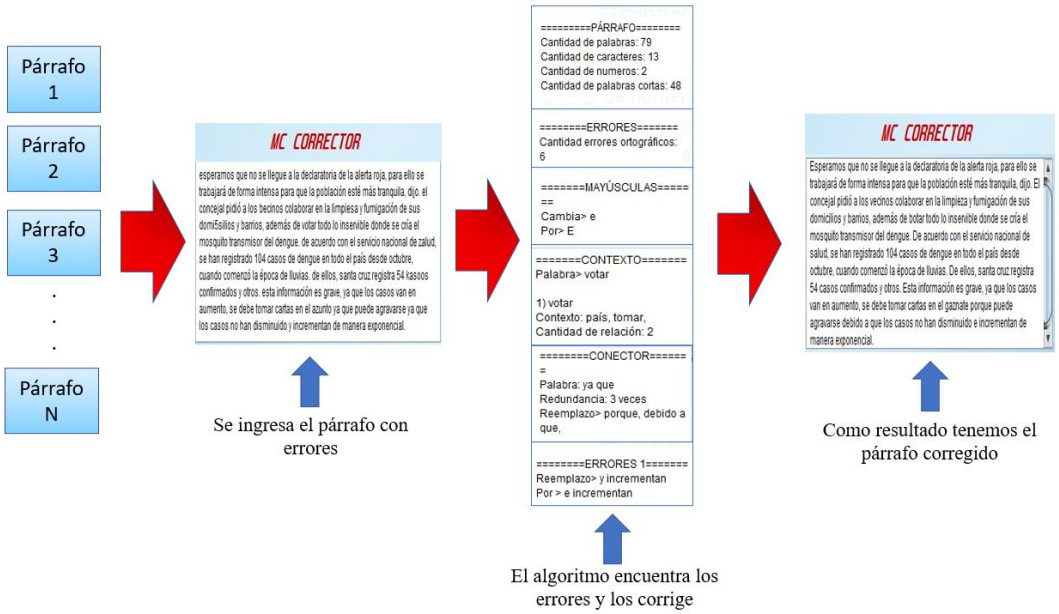


Figura 1 *Modelo de proceso de datos*

Para poder clasificar correctamente si los resultados obtenidos son favorables o desfavorables se obtiene la cantidad de errores corregidos y no corregidos por MC Corrector, según el tipo de error. Para ello se debe considerar lo siguiente (*ver tabla 1*):

- Cantidad de errores ortográficos corregidos.
- Cantidad de errores ortográficos no corregidos.
- Cantidad de errores gramaticales corregidos.
- Cantidad de errores gramaticales no corregidos.
- Cantidad de errores de estilo corregidos.
- Cantidad de errores de estilo corregidos.

En consecuencia, se hace el correcto llenado en la hoja de Excel que nos servirá como instrumento de medición con sus respectivas ubicaciones de los datos obtenidos anteriormente. *Ver tabla 2*

Tabla 2 *Cantidad de errores corregidos y no corregidos*

Párrafos	Errores								
	ortográficos			gramaticales			de estilo		
	Cantidad	MC Corrector		Cantidad	MC Corrector		Cantidad	MC Corrector	
		Corregido	No Corregido		Corregido	No Corregido		Corregido	No Corregido
1	9	9	0	1	1	0	1	1	0
2	5	5	0	1	1	0	0	0	0
3	4	4	0	4	4	0	1	1	0
4	4	4	0	1	1	0	0	0	0
5	8	8	0	1	1	0	0		0
6	4	4	0	1	1	0	0	0	0
7	3	3	0	1	1	0	0	0	0
. . . n	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
TOTAL	Σ(TE)	Σ(EC)	Σ(ENC)	Σ(CE)	Σ(EC)	Σ(ENC)	Σ(CE)	Σ(EC)	Σ(ENC)

Dónde: CE (Total errores), EC (errores corregidos) y ENC (Errores no corregidos)

Paso 5: Obtener la calidad de corrección

Se usará un cuadro comparativo porque es una herramienta que permite comparar resultados de cantidad y el porcentaje de esta. En la tabla 3 se muestra el cuadro comparativo que permite calcular los indicadores del porcentaje de errores ortográficos, gramaticales y de estilo corregidos, donde la primera fila muestra el encabezado que servirá como guía para ingresar los datos correspondientes y las filas debajo de esta, muestra las herramientas con sus respectivos resultados de otros corpus similares.

Tabla 3 *Cuadro comparativo de errores corregidos*

Método	ERRORES		TOTAL	% errores Corregidos
	Corregidos	No corregidos		
Herramienta A	Σ(EC)	Σ(ENC)	Σ(EC) + Σ(ENC)	$\%EC = \frac{\Sigma(\mathrm{EC})}{Total} x\ 100$
Herramienta B	Σ(EC)	Σ(ENC)	Σ(EC) + Σ(ENC)	$\%EC = \frac{\Sigma(\mathrm{EC})}{Total} x\ 100$
Herramienta C	Σ(EC)	Σ(ENC)	Σ(EC) + Σ(ENC)	$\%EC = \frac{\Sigma(\mathrm{EC})}{Total} x\ 100$

Permite comparar el porcentaje de los indicadores de errores ortográficos, gramaticales y de estilo corregidos.

Paso 6: Fórmula para determinar el incremento del porcentaje de errores corregidos con respecto a los resultados de otros corpus. La fórmula fue adaptada para los fines de esta investigación. *Ver figura 2*

1. Realizar operación de porcentaje de errores ortográficos corregidos

$$\%Errores\ ortográficos\ Corregidos = \frac{Errores\ ortográficos\ Corregidos}{Total\ de\ errores\ ortográficos} x\ 100$$

2. Realizar operación de porcentaje de errores gramaticales corregidos

$$\%Errores\ gramaticales\ Corregidos = \frac{Errores\ gramaticales\ Corregidos}{Total\ de\ errores\ gramaticales} x\ 100$$

3. Realizar operación de porcentaje de errores ortográficos corregidos

$$\%Errores\ de\ estilo\ Corregidos = \frac{Errores\ de\ estilo\ Corregidos}{Total\ de\ errores\ de\ estilo} x\ 100$$

3.6 Método de análisis de datos

Se efectuó un análisis estadístico de promedio simple el cual Hanke y Wichern indican que: “es una técnica adecuada cuando se han estabilizado las fuerzas que generan las series a pronosticar y cuando por lo general, no cambia el entorno donde existe la serie” (2006, p. 105). La información procesada y obtenida por MC Corrector fue insertada en las tablas mostradas anteriormente *(ver tablas 2 y 3)* y se aplicó las fórmulas adaptadas del autor Sotomayor, C. et al., (*ver figura 2*) para los cálculos de los indicadores de la investigación, los cuales fueron tabulados en MS Excel.

$$\frac{\text{N° de errores ortográficos}}{\text{N° de palabras escritas}} \text{ x } 100 = \text{ \% de errores por palabras escritas}$$

Figura 2 *Fórmula para obtener el índice que da cuenta del porcentaje de errores (Sotomayor, C. et al., 2012, p .117)*

De los resultados, obtenidos, se llevó a cabo un análisis comparativo de los resultados obtenidos por otros autores. Nohlen (2020) “es el procedimiento de comparación sistemática de objetos de estudio que, por lo general, es aplicado para llegar a generalizaciones empíricas y a la comprobación de hipótesis” (p. 42).

3.7 Aspectos éticos

Esta investigación se realizó con total honestidad en cuanto a la recolección de datos, y citas de los autores de donde se obtuvo la información necesaria para ensamblar y contextualizar este documento, teniendo en cuenta también el Código de Ética en Investigación de la UCV, el Código de Ética del Colegio de Ingenieros del Perú.

La presenté investigación tuvo como base la transparencia de “identificar correctamente y elaborar las referencias bibliográficas correspondientes a distintos tipos de documentos, de acuerdo a la Norma ISO 690:2010” (Alicante,

2008, p. 2) respetando las ideas de otros autores y citándolos debidamente, sin hacer hurto de los derechos de autor, cumpliendo con el código de ética de la investigación de la Universidad Cesar Vallejo (2020) según el artículo 9° de la política anti plagio, el en cual indicó: “Cumplir con los requisitos propuestos por la escuela académica profesional de ingeniería de la Universidad César Vallejo” (p. 9). Además, en el artículo 10, como lo explican las autoridades de la Universidad César Vallejo (2020) los derechos de los autores son un estándar principal en una investigación (p. 10).

Así mismo, en los artículos 37°, 42° y 44° del Código de Ética del Colegio de Ingenieros del Perú (2018) “involucraron la divulgación de información o la omisión de uno o más coautores que participan en la investigación” (p. 46). Por otro lado, se dispuso de un docente con una profunda ética, moral y justicia para el cumplimiento de los fines y objetivos de la universidad, pues “no se debe olvidar que un investigador antes de ser hombre de ciencia, es una persona que cuenta con una formación ética y moral, la cual ha ido adquiriendo a lo largo de su vida” (Morales, Nava, Esquivel y Díaz, 2011, p. 25). Puesto que, “si la sociedad está enferma moralmente, esta pueda contagiar a los investigadores y los científicos” (Ñaupas et al., 204, p. 462).

IV. RESULTADOS

En este capítulo se muestra y explica los resultados obtenidos dentro de la investigación de acuerdo con los indicadores desarrollados como: porcentaje de errores ortográficos corregidos, porcentaje de errores gramaticales corregidos, porcentaje de errores de estilo corregidos. Así mismo, se demostró la efectividad de los algoritmos combinados para la realización de MC Corrector, se comprobó que el algoritmo MC Corrector incrementó el porcentaje de errores ortográficos corregidos e incrementó el porcentaje de errores de estilo corregidos, sin embargo, no se pudo comprobar el porcentaje de incremento de errores gramaticales corregidos, puesto que en la comparación realizada postest se obtuvo el mismo resultado con un algoritmo desarrollado por otro investigador.

Puesto que nuestra investigación solo ha hecho pruebas postest no se han realizado, las pruebas descriptivas o de normalidad por cada indicador mencionado, por lo que pasaremos directamente a las pruebas de hipótesis.

4.1 Prueba de Hipótesis

En esta sección se muestra las pruebas realizadas para cada indicador y se concluye un resultado de las hipótesis.

4.1.1 Prueba de Hipótesis 1

$HE1_0$: El algoritmo MC Corrector no incrementó el porcentaje de errores ortográficos corregidos por lo menos en un 73%.

$HE1_1$: El algoritmo MC Corrector incrementó el porcentaje de errores ortográficos corregidos por lo menos en un 73%.

Tabla 4 *Porcentaje de errores ortográficos corregidos*

Método	ERRORES		TOTAL	Porcentaje de errores Ortográficos Corregidos (%)
	Corregidos	No corregidos		
Base	33	12	45	73.3333
Propuesto	32	13	45	71.1111
Google Docs	18	27	45	40
Word 2016	30	15	45	66.6667
MC corrector	555	36	591	93.9086

Con los resultados obtenidos podemos observar que, a diferencia de las otras aplicaciones de corrección, MC Corrector tiene un porcentaje mayor, lo cual indica que el algoritmo desarrollado tiene una mejor corrección ortográfica, por ende, se rechaza la hipótesis nula $HE1_0$ y se acepta la hipótesis alternativa $HE1_1$.

4.1.2 Prueba de Hipótesis 2

$HE1_0$: El algoritmo MC Corrector no incrementó el porcentaje de errores gramaticales corregidos por lo menos en un 100%.

$HE1_1$: El algoritmo MC Corrector incrementó el porcentaje de errores gramaticales corregidos por lo menos en un 100%.

Tabla 5 *Porcentaje de errores gramaticales corregidos*

METODO	ERRORES		TOTAL	Porcentaje de errores Gramaticales Corregidos (%)
	Corregidos	No Corregidos		
Word	1	8	9	25
SpanishChecker	4	5	9	37
Stilus	3	6	9	23
CorrectMe	9	0	9	100
MC corrector	208	0	208	100

Con los datos obtenidos podemos observar que, a diferencia de las otras aplicaciones de corrección, CorrectMe y MC Corrector tienen un porcentaje mayor y a su vez son iguales. Sin embargo, al cumplir con una corrección de por lo menos un 100% se acepta la hipótesis alterna $HE1_1$ y se rechaza la hipótesis nula $HE1_0$.

4.1.3 Prueba de Hipótesis 3

$HE1_0$: El algoritmo MC Corrector no incrementó el porcentaje de errores de estilo corregidos.

$HE1_1$: El algoritmo MC Corrector incrementó el porcentaje de errores de estilo corregidos.

Tabla 6 *Porcentaje de errores de estilo corregidos*

METODO	ERRORES		TOTAL	Porcentaje de errores de estilo corregidos (%)
	Corregidos	No Corregidos		
Word	13	1	14	92.8571
Stilu	24	3	27	88.8889
MC corrector	84		84	100

Con los resultados obtenidos podemos observar que, a diferencia de las otras aplicaciones de corrección, MC Corrector tiene un porcentaje mayor, lo cual indica que el algoritmo desarrollado tiene una mejor corrección de estilo, por ende, se rechaza la hipótesis nula $HE1_0$ y se acepta la hipótesis alternativa $HE1_1$.

4.2 Resumen

En la tabla 6 se muestra los resultados de aceptaciones o rechazos de las hipótesis planteadas en la investigación.

Tabla 7 *Resumen de los resultados de las pruebas de hipótesis*

Cod.	Hipótesis	Condición
HE1	El algoritmo de corrección de la escritura de textos en español basado en los algoritmos de Metaphone y Distancia de Levenshtein (algoritmo MC Corrector) incrementó el porcentaje de errores ortográficos corregidos por lo menos en un 73%	Aceptada
HE2	El algoritmo MC Corrector incrementó el porcentaje de errores gramaticales corregidos por lo menos en un 100%	Aceptada
HE3	El algoritmo MC Corrector incrementó el porcentaje de errores de estilo corregidos por lo menos en un 92%	Aceptada

V. DISCUSIÓN

En este capítulo se discute acerca de los resultados obtenidos en la presente investigación y los resultados de antecedentes de otras investigaciones que han usado e implementado los algoritmos de Distancia de Leveshtein y Metaphone en sus sistemas de corrección de textos en un enfoque ortográfico, gramatical y de estilo. Además, en los indicadores asociados a las comprobaciones de hipótesis se comparó la calidad de corrección entre MC Corrector y otros sistemas, tales como Word2016, Stilus, SpanishChecker, entre otros.

Los porcentajes de errores ortográficos corregidos con los algoritmos Base (Distancia de Levenshtein), Propuesto (Alteración de costos de Distancia de Levenshtein), Google Docs, Word 2016 y MC Corrector fueron, 73.3333%, 71.1111%, 40%, 66.6667% y 93.9086% respectivamente. Como se puede apreciar, el porcentaje de errores ortográficos corregidos por el algoritmo MC Corrector fue mayor al porcentaje de los otros algoritmos ya mencionados.

Así mismo los resultados obtenidos de la presente investigación son similares a las de Moyotl-hernández (2016) quienes trabajaron con el algoritmo de Distancia de Levenshtein alterando sus costos de distancia y aunque no lograron su objetivo de superar las pruebas, puesto que el algoritmo original obtuvo mejores resultados, llegaron a la conclusión que, para ser un corrector de propósito general obtuvieron buenos resultados. Situación que nos facilitó al tomar el algoritmo original como tal y razón por la cual obtuvimos mejores resultados.

Los porcentajes de los errores gramaticales corregidos por las herramientas, tales como: Word SpanishChecker, Stilus, CorrectMe y MC Corrector fueron 25%, 37%, 23%, 100% y 100% respectivamente. Como se puede observar, el porcentaje del algoritmo MC Corrector fue mayor a Word, SpanishChecker y Stilus, sin embargo, fue igual al algoritmo de CorrectMe., además cabe recalcar que las pruebas realizadas con MC corrector se basaron en párrafos y no en oraciones o pequeñas frases como lo hizo CorectMe.

Como podemos observar los resultados obtenidos en nuestra investigación son similares a los de San Mateo (2016), pero en contraste con este, el algoritmo CorrectMe desarrollado por el autor mencionado se basa en el análisis estadístico, mientras que el algoritmo MC Corrector desarrollado en la

presente investigación se basa en los fonemas y la distancia entre dos palabras. Por otro lado, tenemos a Castro (2015), quien en su investigación hace uso de 5 métodos, siendo 2 de ellos, Metaphone y Distancia de Levenshtein teniendo como conclusiones que, aunque sus resultados son favorables en un 15% entre el corrector Huspell, tiene un 5% de resultados desfavorables con las herramientas de office, lo cual no ocurre con los algoritmos CorrectMe y MC Corrector.

Con respecto a los resultados obtenidos de los porcentajes de errores de estilo corregidos entre las herramientas Word, Stilu y MC Corrector, obtuvieron un 93.8571%, 89.8889% y 100% respectivamente. Como se puede observar, el porcentaje de errores corregidos por el algoritmo MC Corrector es mayor, pese a que Word tiene más tiempo en el mercado vemos una diferencia significativa en la corrección de estilo, basándonos en pruebas realizadas de trabajos de investigación previos.

Así también, se debe tener en cuenta que ante las diferentes correcciones de estilo hay uno que también debería ser incluido puesto que no se toma en cuenta y lo toman como detector sin corregir el problema, pues se ha realizado una búsqueda exhaustiva de algoritmos y herramientas que lleven a cabo la corrección de conectores lógicos repetitivos en un solo párrafo (muletillas) que empobrecen el estilo de redacción (*ver tabla 11*), donde solo se ha encontrado una herramienta “AntConc” mencionada por Quintero (2015) que le fue de utilidad para llevar a cabo un análisis de los conectores lógicos más usados y repetitivos. Sin embargo, no hace corrección alguna ni brinda ayuda, solo es usada como detector. Por lo que podemos decir que nuestro algoritmo es superior ante dicha herramienta ya que, así como detecta, también corrige.

VI. CONCLUSIONES

Las conclusiones de esta investigación fueron las siguiente:

1. Se presentó un algoritmo de la corrección de escritura de textos en español basados en los algoritmos Metaphone y Distancia Levenshtein (algoritmo MC Corrector), donde se indica la efectividad en el porcentaje de errores ortográficos corregidos, porcentaje de errores gramaticales corregidos y porcentaje de errores de estilo corregidos, demostrando así la mejora de resultados en la combinación de estos algoritmos y la importancia que tiene en la corrección de párrafos en base al contexto al no encontrarse investigaciones sobre estos tipos de errores, siendo así uno de los errores más frecuentes.

2. Se realizó un software en java para implementar el algoritmo MC Corrector para la corrección de errores ortográficos, gramaticales y de estilos, en el idioma español y demostrar su efectividad. En base a ello, el software muestra la cantidad de errores ortográficos, gramaticales y de estilos, como también se implementó las correcciones de espacios innecesarios y mayúsculas, pero no se toma como dato relevante en las pruebas.

3. Se observó que el algoritmo MC Corrector incrementó el porcentaje de errores ortográficos corregidos en comparación a otros métodos de corrección de texto en español, dando un resultado de 93% de errores ortográficos corregidos, superando el 73% del corrector base. En base a aquello se demuestra el efecto positivo en los erros ortográficos corregidos utilizando el algoritmo MC Corrector.

4. Con respecto a los resultados del estudio, el algoritmo MC Corrector incrementó el porcentaje de errores gramaticales corregidos, puntualmente los errores contextuales, obteniendo un valor de 100% al igual que el método CorrectMe, sin embargo, cabe destacar que, en esta investigación, se utilizó una mayor cantidad de párrafos (100) en español para realizar las pruebas a comparación de CorrectMe que utiliza solo oraciones cortas para su proceso, siendo así que las pruebas de

CorrectMe fueron con 9 palabras con errores gramaticales, y en las pruebas de esta investigación fueron de 208 palabras con los mismos tipos de errores.

5. Se observó que el algoritmo MC Correcto incrementó el porcentaje de errores de estilo corregidos, obteniendo un valor de 100% demostrando totalidad de la corrección de 100 párrafos con 84 errores de estilos, a diferencia del método Stilu que obtuvo un porcentaje de errores de estilos corregidos de 88% y Word con un 92%.

6. En resumen, de acuerdo a los resultados finales obtenidos se puede concluir que la unión del algoritmo Metaphone, Distancia de Levenshtein y MC Corrector, tuvo un efecto positivo en el porcentaje de los errores ortográficos, gramaticales y de estilos corregidos, demostrando así una mejora significativa.

VII. RECOMENDACIONES

Las recomendaciones para futuras investigaciones son las siguientes:

1. Ampliar la investigación científica con una muestra más amplia de documentos a evaluar. Por ejemplo, párrafos en documentos, periódicos, revistas, tesis, páginas web, etc. con los tipos de errores textuales más frecuentes, ya que las pruebas se realizaron con una muestra de 100 párrafos extraídos del sitio web de la Real Academia Española y fueron alterados con errores voluntarios para hacer las pruebas.
2. Profundizar el análisis de los algoritmos de corrección de texto, de manera que se pueda tener un mayor alcance para el desarrollo de nuevos métodos en base a otros algoritmos relacionados, finalmente así se alcanzaría una mejora significativa del conocimiento de las reglas ortográficas, gramaticales, estilo y entre otros.
3. Ampliar la cantidad de criterios de las comparaciones realizadas. Por ejemplo, en nuestra investigación se usaron tres criterios a evaluar: la cantidad de errores ortográficos, gramaticales y de estilos corregidos. Por lo que se sugiere usar otros criterios, como: la precisión, la velocidad y la exactitud.
4. Desarrollar la investigación científica considerando una muestra o el total de la población a estudiar donde se ingresen textos en tiempo real. Por ejemplo, en la presente investigación se tuvo como base párrafos obtenidos del sitio web de la Real Academia Española, por lo cual se sugiere obtener datos de textos ingresados por personas con bajos conocimientos de las reglas ortográficas, gramaticales y de estilo.
5. Desarrollar investigaciones científicas en regiones donde haya un mayor porcentaje de personas con bajo nivel de educación, especialmente

conocimientos bajos en gramática española. Así se obtendrá textos con errores más significativos que ayudará en la evolución de la herramienta de corrección y a su vez beneficiar a la población, especialmente, con la necesidad de expresarse textualmente de manera correcta.

REFERENCIAS

ALICANTE, U. de, 2008. La norma ISO 690:2010(E) [en línea]. vol. 06, pp. 25. DOI: http://werken.ubiobio.cl/html/downloads/ISO_690/Guia_Breve_ISO690-2010.pdf

ALOGLAH, T. M. A. Spelling errors among arab english speakers. *Journal of Language Teaching & Research* [in líne]. Finlandia: Academy Publication, 2018, **9**(4), pp. 746-753 [viewed: 22 october 2019]. ISSN 1798-4769. Available from: http://dx.doi.org/10.17507/jltr.0904.10

ALONSO SERRANO, A., GARCÍA SANZ, L., LEÓN RODRIGO, I., GARCÍA GORDO, B. y RIOS BREA, L. Métodos de investigación de enfoque experimental, *Universidad Nacional Autónoma de México* [en línea]. 2006, pp. 3-33 [Consultado: 19 de septiembre 2021] Disponible en: https://www.postgradoune.edu.pe/pdf/documentos-academicos/ciencias-de-la-educacion/10.pdf

ALTAMEEMY, F. A., & DARADKEH, A. Common Paragraph Writing Errors Made by Saudi EFL Students: Error Analysis. *Theory and Practice in Language Studies* [online]. Finlandia: Academy Publication, 2019, **9**(2), pp. 178-187 [viewed: 27 october 2019]. ISSN 1799-2591. Available from: http://dx.doi.org/10.17507/tpls.0902.07

ARAYA RAMÍREZ, J. El comportamiento de las categorías gramaticales y la precisión léxica en textos orales narrativos y explicativos producidos por escolares en Costa Rica. *Revista Electrónica "Actualidades Investigativas en Educación"* [en linea]. 2011, **11**(3), pp. 1-25 [consulta: 24 de septiembre de 2019]. ISSN 1409-4703. Disponible en: http://www.redalyc.org/articulo.oa?id=44722178007

ARROJO, Maria. Information and the Internet: An Analysisfrom the Perspective of the Science of the Artificia, *Minds & Machine* [online]. Spain: Springer, 2017, **27**(3), pp. 1-24 [viewed: 22 october 2019]. Available from: 10.1007/s11023-016-9413-2

ATHANASELIS, T., BAKAMIDIS, S. and DOLOGLOU, I. An Automatic method revising ill-formed sentences based on N-grams. *Speech Prosody* [online].

German: ISCA Archive, 2006, pp.1-4 [viewed: April 14, 2021]. Avalaible from: http://www.cs.columbia.edu/~julia/papers/TOBI_i92_0867.pdf

BADII, M. H., CASTILLO, J., LANDEROS, J. y CORTEZ, K. Papel de la estadística en la investigación científica. *Innovaciones de Negocios* [en línea]. México: UANL, 2007, **4**(1), pp. 107-145 [consulta: 20 de octubre 2021]. ISSN 1665-9627. Disponible en: http://eprints.uanl.mx/12472/1/A5.pdf

CHAN, K., VASARDANI, M., and WINTER, S. Getting Lost in Cities: Spatial Patterns of Phonetically Confusing Street Names, *Transactions in GIS* [online]. New York: John Wiley & Sons Ltd, 2015, **19**(4), pp. 535-562. [viewed: 16 october 2019]. ISSN 1361-1682 Available from: https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=109135624&lang=es&site=eds-live

CHRISTANTI, V., RUDY and NAGA, D. Fast and Accurate Spelling Correction Using Trie and Damerau-levenshtein Distance Bigram, *TELKOMNIKA* [online]. Yogyakarta: Universitas Ahmad Dahlan, 2018, **16**(2), pp. 827-833. [viewed: 29 october 2019]. ISSN 1693-6930 Available from: http://dx.doi.org/10.12928/telkomnika.v16i2.6890

COLEGIO DE INGENIEROS DEL PERÚ. Código de ética del colegio de ingenieros del Perú. Código de Ética del CIP, 26. 1999. Extraído de: http://www.cip.org.pe/publicaciones/reglamentosCNCD2018/codigo_de_etica_del_cip.pdf.

CRUZ DEL CASTILLO, C., OLVIRES, S., GONZÁLES, M. *Metodología de la investigación* [en línea]. México D.F.: Grupo Editorial Patria, 2014 [Consultado: 06 de noviembre de 2019]. ISBN 978-607-438-876-3 (Online). Disponible en: https://editorialpatria.com.mx/pdffiles/9786074381498.pdf

DASHTI, SM. Real-word error correction with trigrams: correcting multiple errors in a sentence. *Language Resources & Evaluation* [online]. Iran: Springer, 2017, **52**, pp. 485-502. [viewed: 11 september 2019]. ISSN 1574-020X Available from: https://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=129593482&lang=es&site=eds-live

DRIAS, Y., KECHID, S. & PASI, G. Bee Swarm Optimization for Medical Web Information Foraging. *J Med Syst* [online]. Italy: Springer, 2016, **40**(2), pp. 1-17 [viewed: 15 september 2019]. ISSN 0148-5598 Available from: https://search.ebscohost.com/login.aspx?direct=true&db=edsgih&AN=edsgcl.435255595&lang=es&site=eds-live

DRONEN, N. *Correcting Writing Errors with Convolutional Neural Networks*. Tesis doctoral inédita. M. James H (dir.) y F. Peter W. University of Colorado in Boulder, 2016

Eliminar repeticiones de los textos. En: *Instituto Vasco de Administración Pública [IVAP]* [en línea]. Vasco: Instituto Vasco de Administración Pública [IVAP], 2019. [Consulta: 20 de agosto de 2020]. Disponible en: https://www.ivap.euskadi.eus/entrada-blog/2019/eliminar-repeticiones-de-los-textos/z16-b2aha/es/

EVANGELISTA, D. La cohesión y mecanismos de cohesión en la composición de textos. *Revista Lengua y Sociedad* [en línea]. 2014, **14**(14), pp. 227-244 [Consulta: 02 de septiembre de 2019]. Disponible en: http://revista.letras.unmsm.edu.pe/index.php/ls/article/view/464/435

FAILI, H., NAVA, E., MORTAZA, M. AND MOHAMMAD, T. Vafa spell-checker for detecting spelling, grammatical, and real-word errors of Persian language. *Digital Scholarship in the Humanities* [online]. 2016, **31**(1), pp. 95-117 [viewed: 05 september 2019]. ISSN 2055-768X. Available from: https://academic.oup.com/dsh/article-abstract/31/1/95/2605365

FERNÁNDEZ JODAR, R. Análisis de errores léxicos, morfosintácticos y gráficos en la lengua escrita de los aprendices polacos de español, *Biblioteca virtual redELE* [en línea]. 2006, 8, pp. 8-213 [consultado: 16 de noviembre de 2021]. Disponible en: http://hdl.handle.net/11162/76535

FERNÁNDEZ JÓDAR, R. *ANÁLISIS DE ERRORES LÉXICOS, MORFOSINTÁCTICOS Y GRÁFICOS EN LA LENGUA ESCRITA DE LOS APRENDICES POLACOS DE ESPAÑOL* [en línea]. Tesis Doctoral. Universidad Adam Michiewicz, 2007 [consultado: 21 de noviembre de 2020]. Disponible en:

https://redined.mecd.gob.es/xmlui/bitstream/handle/11162/76535/00820103007276.pdf?sequence=1

FERREIRA, A. y HERNÁNDEZ, S. Diseño e implementación de un corrector ortográfico dinámico para el sistema tutorial inteligente, ELE-TUTORA. *Revista signos.* [en línea]. Valparaíso: Pontificia Universidad Católica, 2017, **50**(95), pp. 385-407 [Consulta: 08 de septiembre 2019]. ISSN 0718-0934. Disponible en: http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-09342017000300385&lng=en&tlng=en

FLORES-RUIZ, E., MIRANDA-NOVALES, MG., VILLASÍS-KEEVER, MÁ. El protocolo de investigación VI: cómo elegir la prueba estadística adecuada. Estadística inferencial. *Revista Alergia México* [en línea]. 2017, **64**(3), pp. 364-370 [Consulta: 02 de septiembre 2019]. ISSN 0002-5151. Disponible en: https://doi.org/10.29262/ram.v64i3.304

GALBIATI RIESCO, J. M. Conceptos básicos de estadística [en línea]. 2015. Disponible en: http://www.jorgegalbiati.cl/ejercicios_4/ConceptosBasicos.pdf

GAMBOA GRAUS, M. E. Estadística aplicada a la investigación educativa. *Revista Dilemas Contemporáneos: Educación, Política y Valores* [en línea]. 2017, **2**(5), pp. 1-32 [Consulta: 25 de septiembre 2019]. ISSN 2007-7890. Disponible en: https://search.ebscohost.com/login.aspx?direct=true&db=eue&AN=127823753&lang=es&site=eds-live

GIRAUD-CARRIER, C., GOODLIFFE, J., JONES, B., and CUEVA, S. Effective record linkage for mining campaign contribution data. Knowledge & Information Systems [online]. New York: Springer, 2015, 45(2), pp. 389-416 [viewed: 21 september 2019]. ISSN 0219-1377. Available from: https://search.ebscohost.com/login.aspx?direct=true&db=iih&AN=109541670&lang=es&site=eds-live

HANKE, J. E. y WICHERN, D. W. *Pronósticos en los negocios, Octava edición* [en línea], México: Pearson Educación, 2006 [Consultado: 15 de julio 2021]. ISBN 970-26-0759-0 (Online). Disponible en:

https://www.academia.edu/37476243/LIBRO_Pronosticos_en_los_negocios_John_E_Hanke_8va_Ed_pdf

HO, TL., OH, S.-R., and KIM, HJ. A parallel approximate string matching under Levenshtein distance on graphics processing units using warp-shuffle operations. *PLoS ONE* [online]. China: Quan Zou, Tianjin University, 2017, **12**(10), pp. 1-15 [viewed: 27 september 2019]. ISSN 1923-6203. Available from: https://doi.org/10.1371/journal.pone.0186251

IGLESIAS HERNÁNDEZ, T. D., GONZALES VALDÉS, A. y HERNÁNDEZ RIVERA, D. L. La progresión temática y la coherencia como criterios textuales en la construcción de párrafos. *Mendive* [en línea]. 2019, **17**(2), pp. 293-309 [Consulta: 25 de septiembre 2019]. ISSN 1815-7696. Disponible en: https://mendive.upr.edu.cu/index.php/MendiveUPR/article/view/1509

KHATTER, S. An Analysis of the Most Common Essay Writing Errors among EFL Saudi Female Learners (Majmaah University). *Arab World English Journal* [online]. Seri Wangsa: Arab Society of English Language Studies, 2019, **10**(13), pp. 364-281 [viewed: 06 september 2019]. ISSN 2229-9327. Available from: https://dx.doi.org/10.24093/awej/vol10no3.26

LAI, K. H., TOPAZ, M., GOSS, F. R., & ZHOU, L. Automated misspelling detection and correction in clinical free-text records. *Journal of Biomedical Informatics* [online]. Boston: Elsevier, 2015, **55**, pp.185-195 [viewed: 13 september 2019]. ISSN 1532-0464. Available from: https://doi.org/10.1016/j.jbi.2015.04.008

Lawley, J. New software to help EFL students self-correct their writing. *Language Learning & Technology* [online]. 2015, **19**(1), pp. 23-33 [viewed: 06 september 2019]. ISSN 1094-3501. Available from: https://scholarspace.manoa.hawaii.edu/bitstream/10125/44396/1/19_01_action1.pdf

LHOUSSAIN, A., HICHAM, G. & ABDELLAH, Y. Adaptating the Levenshtein distance to contextual spelling correction. *International Journal of Computer Science & Applications* [online]. 2015, **19**(1), pp. 127-133 [viewed: 25 september 2019]. ISSN 0972-9038. Available from:

https://search.ebscohost.com/login.aspx?direct=true&db=egs&AN=116745898&lang=es&site=eds-live

MONJE, C. Metodología de la investigación cuantitativa y cualitativa. Guía didáctica [en línea]. 2011. Disponible en: https://www.uv.mx/rmipe/files/2017/02/Guia-didactica-metodologia-de-la-investigacion.pdf

MONTIEL MONTIEL, J. P. *Detección y corrección de errores basados en reglas gramaticales del inglés en conjunto con el mejoramiento de estilos de escritura, aplicado en artículos científicos mediante Algoritmos de Procesamiento de Lenguaje Natural (NLP)* [en línea]. Pontificia universidad católica de Valparaíso, 2018 [consultado: 26 de noviembre de 2019]. Disponible en: http://opac.pucv.cl/pucv_txt/txt-8000/UCC8107_01.pdf

MORALES LIZARAZO, E. *La Recolección de Datos* [diapositiva]. 2014 [Consultado: 23 de enero de 2020]. Disponible en: https://es.slideshare.net/edimor72/la-recoleccin-de-datos-1384547

MORALES, J., NAVA, G., ESQUIVEL, J. and DIAZ, L. Principios de ética, Bioética y conocimiento del hombre. 2011, México: Universidad Autónoma del Estado de Hidalgo.

MOSAVI MIANGAH, T. FarsiSpell: A spell-checking system for Persian using a large monolingual corpus. *Literary and Linguistic Computing* [online]. United Kingdom: Digital Scholarship in the Humanities, 2014. pp. 56-73. [viewed: October 18, 2020]. ISSN 0268-1145. Avalaible from: https://doi.org/10.1093/llc/fqu003

MOUKRIM, C., TRAGHA, A., BENLAHMER, E. H., & ALMALKI, T. An innovative approach to autocorrecting grammatical errors in Arabic texts. *Journal of King Saud University - Computer and Information Sciences* [online]. Morocco: Elsevier, 2019, pp. 476-488 [viewed: 17 october 2019]. ISSN 1319-1578. Avalaible from: https://reader.elsevier.com/reader/sd/pii/S1319157818310012?token=89FE3990E62A42C681E2746389306FCCA13F8914FA28C7188D1BC135F185B0F0A8BD03FBD86846A1B6F1A88DFC7537DD

MOYOTL-HERNÁNDEZ, E. Método de corrección ortográfica basado en la frecuencia de las letras. *Research in Computing Science* [en línea]. 2016, **124**(1), pp. 145-156 [Consulta: 06 de noviembre de 2019] ISSN 1870-4069. Disponible en: http://dx.doi.org/10.13053/rcs-124-1-12

MUHAMMAD IFTE KHAIRUL ISLAM, MD. TAREK HABIB, MD. SADEKUR RAHMAN, MD. RIAZUR RAHMAN and FARRUK AHMED. A Context-Sensitive Approach to Find Optimum Language Model for Automatic Bangla Spelling Correction. *IJACSA:International Jounal of Advance Computer Science and Applicactions* [online]. Dhaka: The Science and Information (SAI) Organization. 2018, **9**(11), pp. 184-191 [viewed: 30 october 2019]. Avalaible from: https://dx.doi.org/10.14569/IJACSA.2018.091126

NOHLEN, D. Capítulo tercero el método comparativo, Biblioteca Jurídica Virtual [en línea]. México DC.: DR., 2020, pp. 42-57 [consultado: 24 de octubre de 2021]. Disponible en: https://archivos.juridicas.unam.mx/www/bjv/libros/13/6180/5.pdf

ÑAUPAS, Humberto; Mejía, Elías y VILLAGÓMEZ Alberto. Metodología de investigación cuantitativa - cualitativa y redacción de tesis. 4ª ed. Ediciones de la U, 2014. 538 pp. ISBN: 9789587621884.

RODRÍGUEZ ARAÍNGA, W. *Guía de Investigación Científica* [en línea]. Lima: Fondo Editorial UCH, 2011 [Consultado: 25 de agosto 2021]. ISBN 978-612-4109-04-1 (Online). Disponible en: https://repositorio.uch.edu.pe/bitstream/handle/20.500.12872/23/rodriguez_arainaga_walabonso_guia%20_investigacion_cientifica.pdf?sequence=1&isAllowed=y

SAN MATEO, A. Un corpus de bigramas utilizado como corrector ortográfico y gramatical destinado a hablantes nativos de español. *Revista Signos* [en línea]. Chile: PUCV, 2016. **49**(90), pp. 94-118. [Consulta: 24 de noviembre de 2020]. ISSN 0718-0934. Disponible en: https://www.scielo.cl/pdf/signos/v49n90/a05.pdf

SANTOS SÁNCHEZ, G. *Validez y confiabilidad del cuestionario de calidad de vida SF-36 en mujeres con LUPUS, Puebla* [en línea]. Tesis de pregrado. Universidad Autónoma de Puebla, 2017 [Consultado: 16 de noviembre de 2021]. Disponible en:

https://www.fcfm.buap.mx/assets/docs/docencia/tesis/ma/GuadalupeSantosSanchez.pdf

Spanish Editors Association [SEA]. Principios profesionales de corrección y edición de textos [en línea]. 2018, pp. 1-15 [Cosulta: 25 de octubre de 2021]. Disponible en: https://spanisheditors.org/resources/Documents/SEA_Principios_profesionales_correcci%C3%B3n_edici%C3%B3n.pdf

UNIVERSIDAD CÉSAR VALLEJO. Código de ética en investigación de la Universidad César Vallejo. 2020, pp. 1-12. Extraído de: https://www.ucv.edu.pe/datafiles/C%C3%93DIGO%20DE%20%C3%89TICA.pdf.

WANG, S. and XU, H. Design of an Intelligent Support System for English Writing Based on Rule Matching and Probability Statistics. *International Journal of Emerging Technologies in Learning (iJET)* [online]. 2018, **13**(11), pp. 157-169 [viewed: July 16, 2021]. Avalaible from: https://online-journals.org/index.php/i-jet/article/view/9608/5292

YULIANTO, M. M., ARIFUDIN, R., & ALAMSYAH, A. Autocomplete and Spell Checking Levenshtein Distance Algorithm To Getting Text Suggest Error Data Searching In Library. *Scientific Journal of Informatics* [online]. 2018. **5**(1), pp. 67-75. [viewed: March 15, 2020]. ISSN 2460-0040. Avalaible from: https://journal.unnes.ac.id/nju/index.php/sji/article/view/67/pdf

ZELASCO, J. F., HOHENDAHL, A. y DONAYO, J. Estado del arte en… Corrección ortográfica automática. *Coordenadas* [en línea]. 2015, pp. 10-16 [Consultado: 20 de diciembre del 2020] Disponible en: https://docplayer.es/52065105-Correccion-ortografica-automatica.html

Anexo 3: Matriz de operacionalización de variables

En la tabla 8 se muestra la matriz de operacionalización de variables donde explica la definición conceptual y operacional de cada variable de esta investigación y a su vez indica la dimensión, indicador, instrumento de medición y escala de medición.

Tabla 8 *Matriz de operacionalización de variables de la investigación*

Variable	Definición conceptual	Definición Operacional	Dimensión	Indicador	Instrumento	Escala de medición
Calidad de corrección	"Su objetivo es garantizar la calidad del lenguaje y la eficacia de la comunicación escrita". (Asociación Española de Editores [SEA], 2018, p. 4)	El algoritmo funcionará bajo el enfoque de realizar una corrección ortográfica, gramatical y de estilo de manera automática.	Calidad de corrección ortográfica	Porcentaje de errores ortográficos corregidos	Hoja de recopilación de datos	%
			Calidad de corrección gramatical	Porcentaje de errores gramaticales corregidos	Hoja de recopilación de datos	%
			Calidad de corrección de estilo	Porcentaje de errores de estilo corregidos	Hoja de recopilación de datos	%

Anexo 4: Matriz de Consistencia

En la tabla 9 se muestra la matriz de consistencia, el cual nos permite tener un mejor enfoque en la construcción de los problemas de la investigación, así como los objetivos e hipótesis generales. También ayuda en la especificación de cada uno de ellos, indicando la variable, sus dimensiones e indicadores.

Tabla 9 *Matriz de consistencia*

PROBLEMAS	OBJETIVO	HIPÓTESIS	VARIABLE	DIMENSIONES	INDICADORES
General	General	General			
¿Cuál fue el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Levenshtein en la corrección de errores ortográficos, gramaticales y de estilo?	Determinar el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Levenshtein en la corrección de errores ortográficos, gramaticales y de estilo.	El algoritmo de corrección de la escritura de textos en español basado en los algoritmos de Metaphone y Levenshtein (algoritmo MC Corrector) incrementó el porcentaje de errores ortográficos, gramaticales y de estilo corregidos.	-	-	-

Específicos	Específicos	Específicos			INDICADORES
¿Cuál fue el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Levenshtein en la corrección de errores ortográficos?	Determinar el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Levenshtein en la corrección de errores ortográficos.	El algoritmo de corrección de la escritura de textos en español basado en los algoritmos de Metaphone y Levenshtein (algoritmo MC Corrector) incrementó el porcentaje de errores ortográficos corregidos. (Moukrim, Tragha, Benlahmer & Almalki, 2019, p. 1)	Calidad de corrección (Asociación Española de Editores [SEA], 2018, p. 4)	Calidad de corrección ortográfica (Moyotl-Hernández, 2016, p. 146)	Porcentaje de errores ortográficos corregidos (Moyotl-Hernández, 2016, p. 146)
¿Cuál fue el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Levenshtein en la corrección de errores gramaticales?	Determinar el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Levenshtein en la corrección de errores gramaticales.	El algoritmo MC Corrector incrementó el porcentaje de errores gramaticales corregidos.		Calidad de corrección gramatical (Moyotl-Hernández, 2016, p. 147)	Porcentaje de errores gramaticales corregidos (Moyotl-Hernández, 2016, p. 147)

¿Cuál fue el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Levenshtein en la corrección de errores de estilo?	Determinar el efecto de un algoritmo de corrección de textos en español basado en los algoritmos Metaphone y Levenshtein en la corrección de errores de estilo.	El algoritmo MC Corrector incrementó el porcentaje de errores de estilo corregidos.		Calidad de corrección de estilo (Moyotl-Hernández, 2016, p. 147)	Porcentaje de errores de estilo corregidos (Moyotl-Hernández, 2016, p. 147)

Anexo 5: Arquitectura Tecnológica

En la figura 3 se muestra la arquitectura tecnológica el cual indica los requerimientos de hardware y software, así también como el proceso desde que el usuario ingresa el texto inicial con los errores ortográficos, gramaticales y de estilo (muletillas), el proceso lógico en el que se muestra la secuencia de algoritmos y por último el proceso final donde se clasifica y muestra el texto corregido.

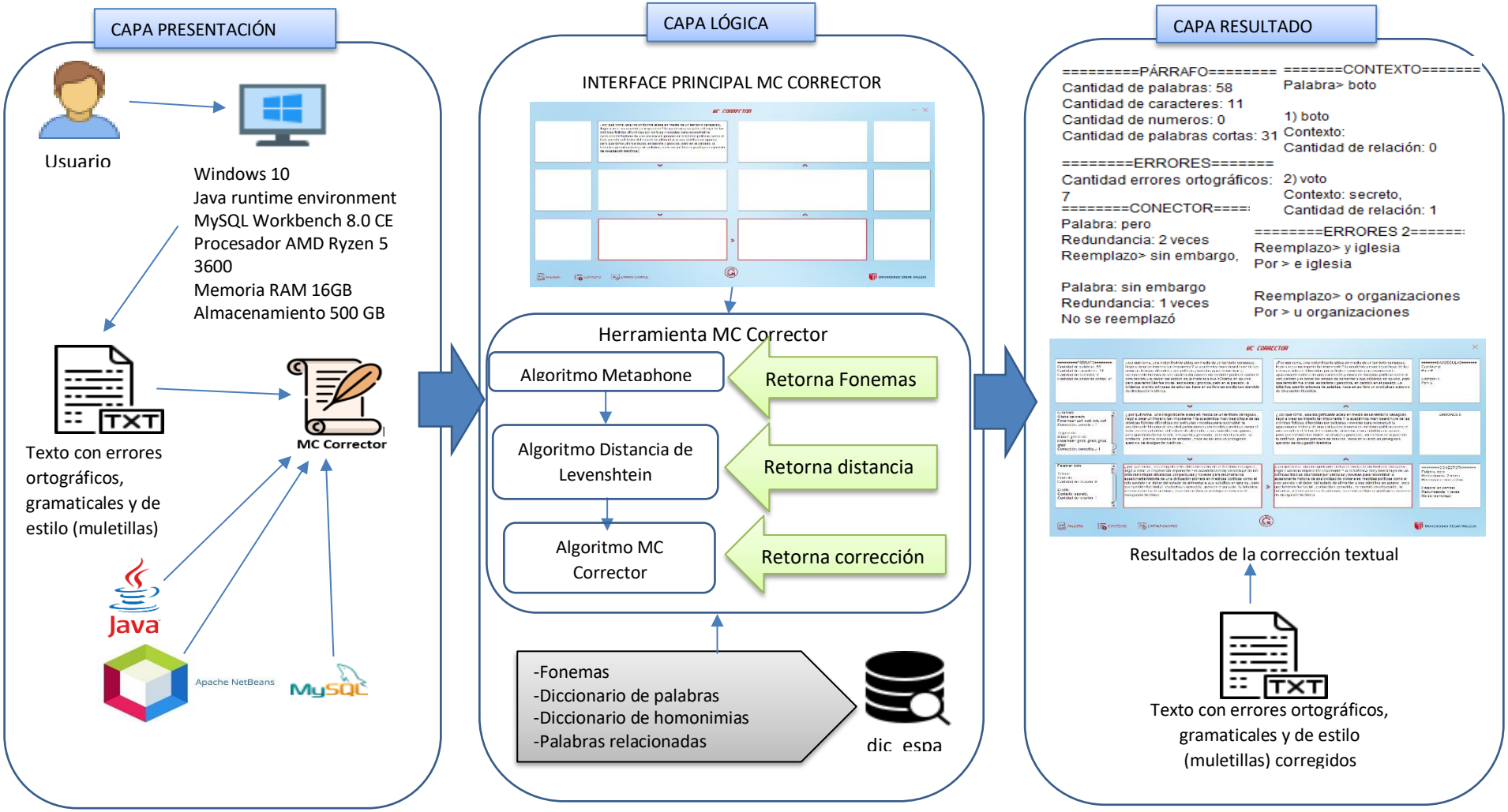


Figura 3 Arquitectura tecnológica

Anexo 6: Proceso principal de la herramienta MC Corrector

El usuario ingresa el párrafo con errores: este debe tener los tipos de errores mencionados en esta investigación (fallos ortográficos, gramaticales y de estilo).

El programa verifica el ingreso de párrafo a corregir: si el programa detecta el campo vacío, este solicitará al usuario que ingrese un párrafo a corregir y reiniciará el proceso. Si la verificación es verdadera, se llamará al método que extrae las palabras de un párrafo y las devuelve en un arreglo.

El programa recorre cada palabra extraída del párrafo, haciendo una consulta a la base de datos por cada una de ellas. Sí la palabra existe, no se corregirá ortográficamente ya que se da a entender que la palabra no está mal escrita. Si la palabra no existe, procede con el siguiente proceso.

El programa llama al método Silabeo, enviando como parámetro principal cada palabra del párrafo. Este devolverá un arreglo con las silabas de cada palabra, en la que su tamaño dependerá de la cantidad de silabas. Por ejemplo, el arreglo de la palabra “zapato”, será de un tamaño de 3, asignados de forma consecutiva: $arreglo_0$=”za”, $arreglo_1$=”pa” y $arreglo_2$=”to”. Esto se realiza para todas las palabras del párrafo.

Cuando todas las palabras tienen sus respectivos arreglos con las silabas indicadas correctamente, se llama al algoritmo Metaphone por cada arreglo de silabas. *Ver Anexo 7: Proceso del algoritmo Metaphone*

Luego de consultar los posibles fonemas, se valida que este haya devuelto resultado, se verifica los fonemas. Si el tamaño de los fonemas es 0, indicar que la palabra no tiene corrección y asignarle el mismo valor de la palabra, de lo contrario, consulta en la base de datos todas las palabras con los fonemas obtenido.

Sí los fonemas consultados en la base de datos no tienen resultados, se indica que la palabra no tiene sugerencias para corrección, de lo contrario, continúa con el paso de la Distancia de Levenshtein para obtener la distancia más corta de operaciones entre la palabra correcta e incorrecta. *Ver Anexo 9: Proceso de algoritmo Distancia de Levenshtein*

Se verifica la distancia más corta entre la palabra correcta y la incorrecta, inicializando la variable distancia con el valor obtenido de Distancia de Levenshtein de cada palabra. Se verifica el índice de la última distancia más corta y se asigna la nueva palabra con el recorrido más corto.

La nueva palabra se va asignando a una variable donde se acumula cada palabra que recorre el bucle, formándose así el nuevo párrafo. Este nuevo párrafo será nuestro texto corregido ortográficamente, pasando así al nuevo algoritmo propuesto MC Corrector para la corrección contextual donde se hará un recorrido de todas las palabras relacionadas en base a su contexto, este proceso retornará un valor entero el cual nos indicará que homonimia es la correcta en base a su cantidad de relación y reemplazará la palabra en el párrafo. *Ver Anexo 10: Proceso del algoritmo MC Corrector*

Se verifica si el párrafo contiene conectores lógicos (*ver Tabla 11 Conectores lógicos textuales)*. Si esto es falso, no se modificará el texto, finalizando así el proceso de corrección. Si tiene conectores lógicos, se hará un recorrido de todos los conectores definidos. Por ejemplo, si se encontró el conector “porque”, generará un bucle de la cantidad de equivalentes que el conector tiene: “porque”, “ya que”, “debido a que”, “dado que”, “pues”, “puesto que”, “por el hecho de que”. Estos datos pueden ser modificados por el usuario, el algoritmo se adapta al tipo de conector y cantidad de equivalentes que se define en un comienzo. *Ver Anexo 10: Proceso del algoritmo MC Corrector*

Por último, se retorna el párrafo corregido en errores de estilo (conectores lógicos) y se realiza una limpieza de caracteres y espacios innecesarios. Así también como proceso final, se añade mayúsculas a la primera letra del Párrafo y al carácter siguiente después de un punto, según la Real Academia Española, mostrando así en la interface del usuario un párrafo limpio de errores ortográficos, gramaticales y de estilo.

A continuación, en la figura 4 se presenta el pseudocódigo del proceso explicado, donde se muestra el flujo principal algorítmico de MC Corrector al ejecutar el software, indicando los nombres de las variables utilizadas, el tipo de variables, las funciones, bucles y condiciones. *Ver figura 4*

```
Algoritmo ejecutarCorrector
    Leer parrafo

    Si Longitud(parrafo)>0 Entonces
        palabras<-sustraerPalabras(parrafo); // metodo que extrae las palabras por espacio
        Para i<-0 Hasta Longitud(palabras) Con Paso 1 Hacer
            Definir palabraCorrecta Como Caracter
            existe<-existePalabra(palabras) //Consulta en la base de datos si existe la Palabra
            Si existe=falso Entonces
                silaba<-silabear(palabras) //devuelve las las silabas de la palabra enviada como parametro
                fonema<-metaphone(silaba) //devuelve todos los posibles fonema de la palabra
                Si Longitud(fonema)>0 Entonces
                    posiblesPalabras<-consultarFonemasBD(fonema) //devuelve las coincidencia de fonemas en la base de datos
                    Definir distancia Como Entero
                    Definir distanciaMasCorta Como Entero
                    Para j<-0 Hasta Longitud(posiblesPalabras) Con Paso 1 Hacer
                        distancia<-levenshteinDistance(palabras,posiblesPalabras) //devuelve el número de operaciones más corto para modificar
                        Si distancia < distanciaMasCorta Entonces
                            palabraCorrecta<-posiblesPalabras
                            distanciaMasCorta<-distancia
                        Fin Si
                    Fin Para
                SiNo
                    Escribir "No tiene fonema"
                    palabraCorrecta<-palabras
                Fin Si
            SiNo
                Escribir "Palabra Correcta"
                palabraCorrecta<-palabras
            Fin Si
            parrafo=+palabraCorrecta
        Fin Para

        parrafo=corregirContexto(parrafo)
        parrafo=CorregirRedundancia(parrafo)
        Escribir parrafo

    SiNo
        Escribir "INGRESAR UN PARRAFO"
    Fin Si

FinAlgoritmo
```

Figura 4 Pseudocódigo principal – Ejecutar corrector

En la figura 5 se visualiza el diagrama de flujo principal del procedimiento explicado donde muestra la secuencia de manera gráfica, en el cual indica los procesos de inicio y fin. *Ver figura 5*

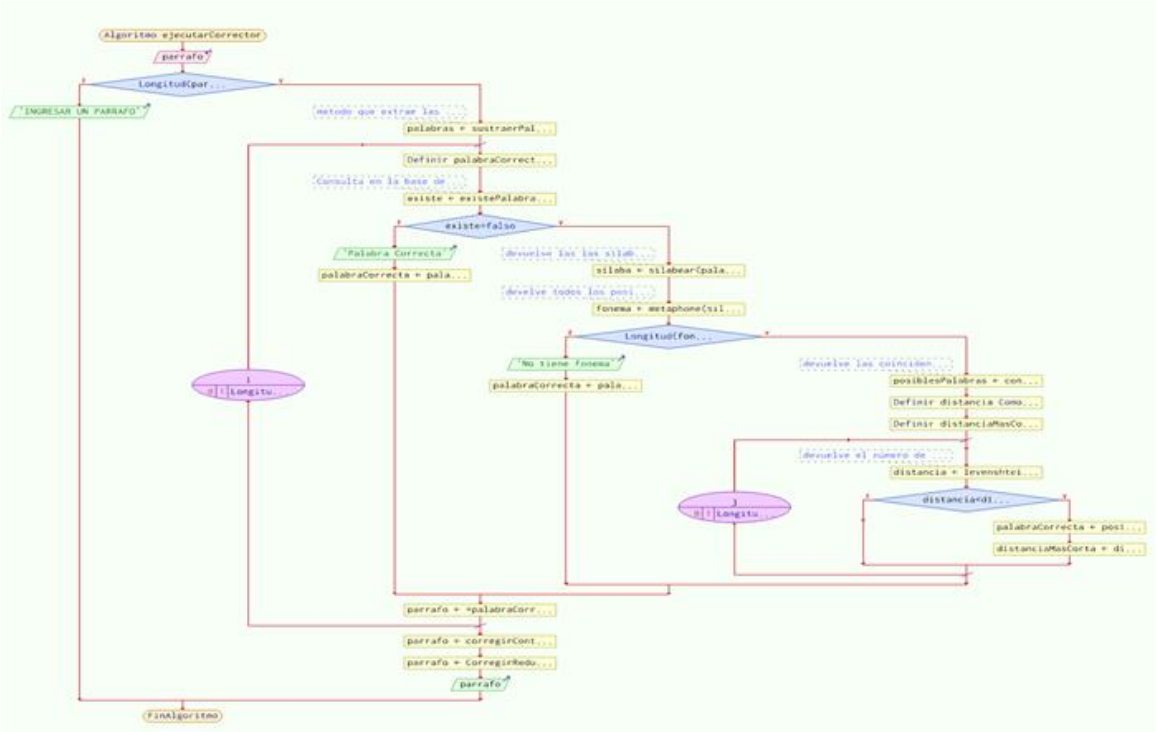


Figura 5 Diagrama de flujo principal – Ejecutar Corrector

Anexo 7: Proceso del algoritmo Metaphone

Cuando todas las palabras tienen sus respectivos arreglos con las silabas indicadas correctamente, se llama al algoritmo Metaphone por cada palabra.

El algoritmo Metaphone es llamado con un parámetro principal, que es el arreglo de silabas de una palabra del párrafo.

Se llama al método para retornar el arreglo de las silabas "retornaArregloSilabas()", este devolverá los fonemas por cada silaba de la palabra. Por ejemplo, sí se envía como parámetro la silaba "se", se retornará los posibles fonemas "se", "ce" y "ze". Las reglas de los fonemas en español para el algoritmo Metaphone se indican en el anexo 15. *Ver Anexo 15: Reglas fonéticas del español.*

Se llama al siguiente método "retornarCombinacionArreglos()" con los posibles fonemas y las silabas, este devolverá la combinación de todos los fonemas de las silabas de la palabra. Por ejemplo, si la palabra es "aseo", fonemas "se", "ce" y "ze", este método retornará "aseo", "aceo" y "azeo". *Ver anexo 8: Estructura de combinación de fonemas.*

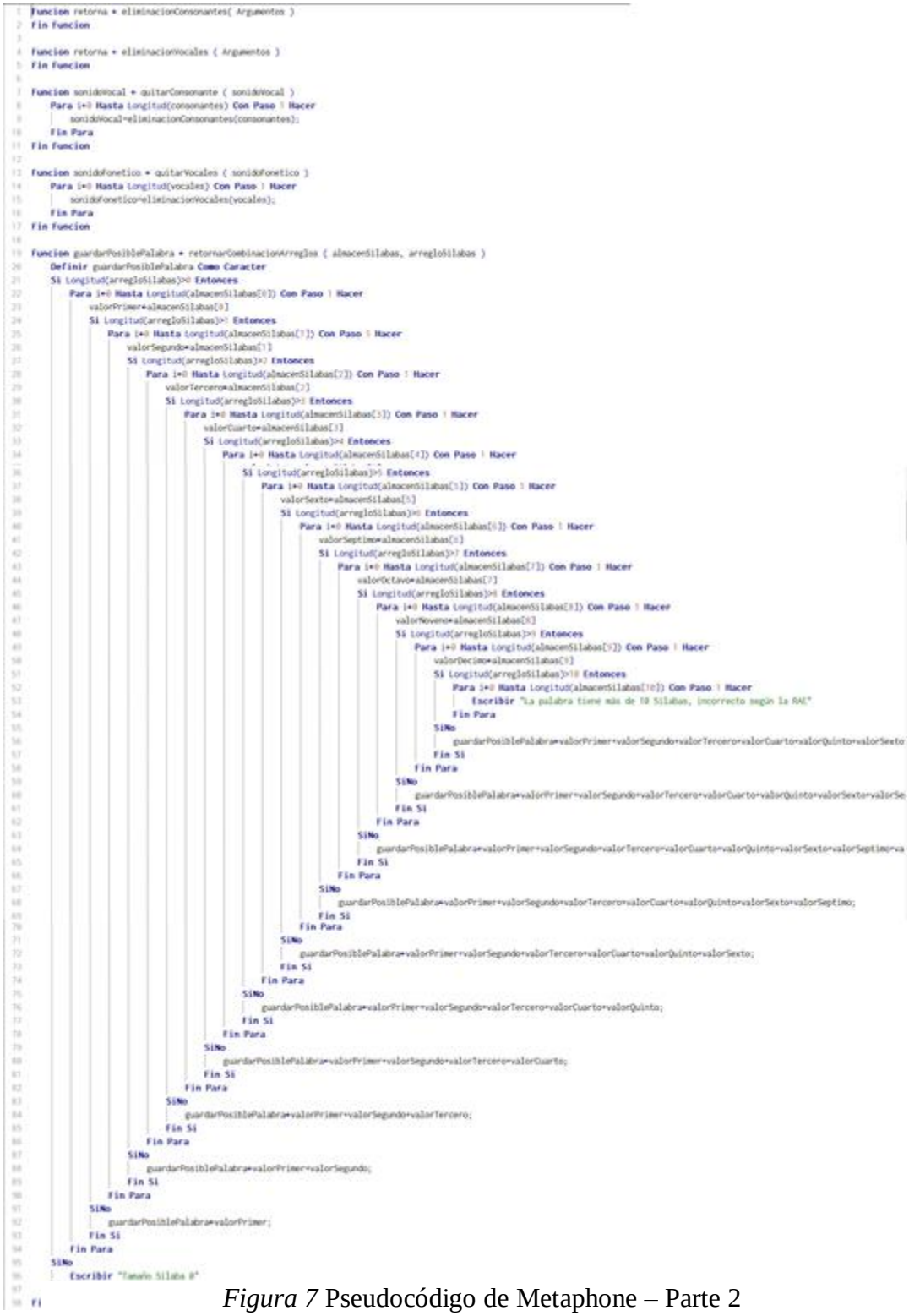
Por último, para obtener los fonemas posibles de la palabra incorrecta, se llama al método "quitarVocales()" dando así los resultados del algoritmo Metaphone.

En la figura 6 se muestra el pseudocódigo de una de las funciones de Metaphone donde retorna los fonemas de cada silaba de las palabras enviadas en su parámetro. *Ver figura 6*

```
Funcion objSilaba ← retornaArregloSilabas (arregloSilabaPalabra)

    Definir objSilaba Como Caracter
    Dimension objSilaba[Longitud(arregloSilabaPalabra)]

    Para i←0 Hasta Longitud(arregloSilabaPalabra) Con Paso 1 Hacer
        sinVocal←quitarVocales(arregloSilabaPalabra[i]);
        sinConsonante←quitarConsonante(arregloSilabaPalabra[i]);
        silabaComienzaConVocal←Falso
        Dimension caracteresSilaba[Longitud(arregloSilabaPalabra[i])]
        caracteresSilaba←arregloSilabaPalabra[i]
        tamanioCaracterSilabas←Longitud(caracteresSilaba[i])

        Para j←0 Hasta Longitud(vocales) Con Paso 1 Hacer
            Si caracteresSilaba[0]=vocales Entonces
                silabaComienzaConVocal=Verdadero
            SiNo
                silabaComienzaConVocal=Falso
            Fin Si
        Fin Para

        Si silabaComienzaConVocal=Falso Entonces
            Si sinVocal="b" Entonces // PARA LA LETRA B
                Dimension letraB[5,cantFonemas] //El constructor asignará los valores a esta variable Vector
                Si sinConsonante="a" Entonces
                    Para j←0 Hasta Longitud(letraB[0,j]) Con Paso 1 Hacer
                        objSilaba[i]=letraB[0,j] //letraB[0][j]-a
                    Fin Para
                Fin Si
                Si sinConsonante="e" Entonces
                    Para j←0 Hasta Longitud(letraB[1,j]) Con Paso 1 Hacer
                        objSilaba[i]=letraB[1,j] //letraB[1][j]-e
                    Fin Para
                Fin Si
                Si sinConsonante="i" Entonces
                    Para j←0 Hasta Longitud(letraB[2,j]) Con Paso 1 Hacer
                        objSilaba[i]=letraB[2,j] //letraB[2][j]-i
                    Fin Para
                Fin Si
                Si sinConsonante="o" Entonces
                    Para j←0 Hasta Longitud(letraB[3,j]) Con Paso 1 Hacer
                        objSilaba[i]=letraB[3,j] //letraB[3][j]-o
                    Fin Para
                Fin Si
                Si sinConsonante="u" Entonces
                    Para j←0 Hasta Longitud(letraB[3,j]) Con Paso 1 Hacer
                        objSilaba[i]=letraB[3,j] //letraB[1][j]-u
                    Fin Para
                SiNo
                    objSilaba[i]=arregloSilabaPalabra[i]
                Fin Si
            SiNo
                Si sinVocal="c" Entonces // PARA LA LETRA C
                    Dimension letraC[5,cantFonemas] //El constructor asignará los valores a esta variable Vector
                    Si sinConsonante="a" Entonces
                        Para j←0 Hasta Longitud(letraC[0,j]) Con Paso 1 Hacer
                            objSilaba[i]=letraC[0,j] //letraC[0][j]-a
                        Fin Para
                    Fin Si
                    //SE APLICA LA MISMA LÓGICA PARA TODAS LAS VOCALES
                SiNo
                    // SE APLICA LA MÍSMA LÓGICA PARA TODAS LAS CONSONANTES (d,f,g,h...)
                Fin Si
            Fin Si
        Fin Si
    Fin Para
Fin Funcion
```

Figura 6 Pseudocódigo de Metaphone – Parte 1

El pseudocódigo de la figura 7 muestra el proceso del proceso del algoritmo Metaphone explicado anteriormente, donde retorna la combinación de todos los posibles fonemas de una palabra. *Ver figura 7*



```
Funcion retorna ← eliminacionConsonantes( Argumentos )
Fin Funcion

Funcion retorna ← eliminacionVocales ( Argumentos )
Fin Funcion

Funcion sonidoVocal ← quitarConsonante ( sonidoVocal )
    Para i←0 Hasta Longitud(consonantes) Con Paso 1 Hacer
        sonidoVocal=eliminacionConsonantes(consonantes);
    Fin Para
Fin Funcion

Funcion sonidoFonetico ← quitarVocales ( sonidoFonetico )
    Para i←0 Hasta Longitud(vocales) Con Paso 1 Hacer
        sonidoFonetico=eliminacionVocales(vocales);
    Fin Para
Fin Funcion

Funcion guardarPosiblePalabra ← retornarCombinacionArreglos ( almacenSilabas, arregloSilabas )
    Definir guardarPosiblePalabra Como Caracter
    Si Longitud(arregloSilabas)>0 Entonces
        Para i←0 Hasta Longitud(almacenSilabas[0]) Con Paso 1 Hacer
            valorPrimer←almacenSilabas[0]
            Si Longitud(arregloSilabas)>1 Entonces
                Para i←0 Hasta Longitud(almacenSilabas[1]) Con Paso 1 Hacer
                    valorSegundo←almacenSilabas[1]
                    Si Longitud(arregloSilabas)>2 Entonces
                        Para i←0 Hasta Longitud(almacenSilabas[2]) Con Paso 1 Hacer
                            valorTercero←almacenSilabas[2]
                            Si Longitud(arregloSilabas)>3 Entonces
                                Para i←0 Hasta Longitud(almacenSilabas[3]) Con Paso 1 Hacer
                                    valorCuarto←almacenSilabas[3]
                                    Si Longitud(arregloSilabas)>4 Entonces
                                        Para i←0 Hasta Longitud(almacenSilabas[4]) Con Paso 1 Hacer
                                            Si Longitud(arregloSilabas)>5 Entonces
                                                Para i←0 Hasta Longitud(almacenSilabas[5]) Con Paso 1 Hacer
                                                    valorSexto←almacenSilabas[5]
                                                    Si Longitud(arregloSilabas)>6 Entonces
                                                        Para i←0 Hasta Longitud(almacenSilabas[6]) Con Paso 1 Hacer
                                                            valorSeptimo←almacenSilabas[6]
                                                            Si Longitud(arregloSilabas)>7 Entonces
                                                                Para i←0 Hasta Longitud(almacenSilabas[7]) Con Paso 1 Hacer
                                                                    valorOctavo←almacenSilabas[7]
                                                                    Si Longitud(arregloSilabas)>8 Entonces
                                                                        Para i←0 Hasta Longitud(almacenSilabas[8]) Con Paso 1 Hacer
                                                                            valorNoveno←almacenSilabas[8]
                                                                            Si Longitud(arregloSilabas)>9 Entonces
                                                                                Para i←0 Hasta Longitud(almacenSilabas[9]) Con Paso 1 Hacer
                                                                                    valorDecimo←almacenSilabas[9]
                                                                                    Si Longitud(arregloSilabas)>10 Entonces
                                                                                        Para i←0 Hasta Longitud(almacenSilabas[10]) Con Paso 1 Hacer
                                                                                            Escribir "La palabra tiene más de 10 Silabas, incorrecto según la RAE"
                                                                                        Fin Para
                                                                                    SiNo
                                                                                        guardarPosiblePalabra←valorPrimer+valorSegundo+valorTercero+valorCuarto+valorQuinto+valorSexto
                                                                                    Fin Si
                                                                                Fin Para
                                                                            SiNo
                                                                                guardarPosiblePalabra←valorPrimer+valorSegundo+valorTercero+valorCuarto+valorQuinto+valorSexto+valorSe
                                                                            Fin Si
                                                                        Fin Para
                                                                    SiNo
                                                                        guardarPosiblePalabra←valorPrimer+valorSegundo+valorTercero+valorCuarto+valorQuinto+valorSexto+valorSeptimo+va
                                                                    Fin Si
                                                                Fin Para
                                                            SiNo
                                                                guardarPosiblePalabra←valorPrimer+valorSegundo+valorTercero+valorCuarto+valorQuinto+valorSexto+valorSeptimo;
                                                            Fin Si
                                                        Fin Para
                                                    SiNo
                                                        guardarPosiblePalabra←valorPrimer+valorSegundo+valorTercero+valorCuarto+valorQuinto+valorSexto;
                                                    Fin Si
                                                Fin Para
                                            SiNo
                                                guardarPosiblePalabra←valorPrimer+valorSegundo+valorTercero+valorCuarto+valorQuinto;
                                            Fin Si
                                        Fin Para
                                    SiNo
                                        guardarPosiblePalabra←valorPrimer+valorSegundo+valorTercero+valorCuarto;
                                    Fin Si
                                Fin Para
                            SiNo
                                guardarPosiblePalabra←valorPrimer+valorSegundo+valorTercero;
                            Fin Si
                        Fin Para
                    SiNo
                        guardarPosiblePalabra←valorPrimer+valorSegundo;
                    Fin Si
                Fin Para
            SiNo
                guardarPosiblePalabra←valorPrimer;
            Fin Si
        Fin Para
    SiNo
        Escribir "Tamaño Silaba 0"

Fi
```

Figura 7 Pseudocódigo de Metaphone – Parte 2

En la figura 8 se muestra el diagrama de flujo del algoritmo Metaphone donde indica la secuencia de manera gráfica la secuencia de cada proceso. *Ver figura 8*

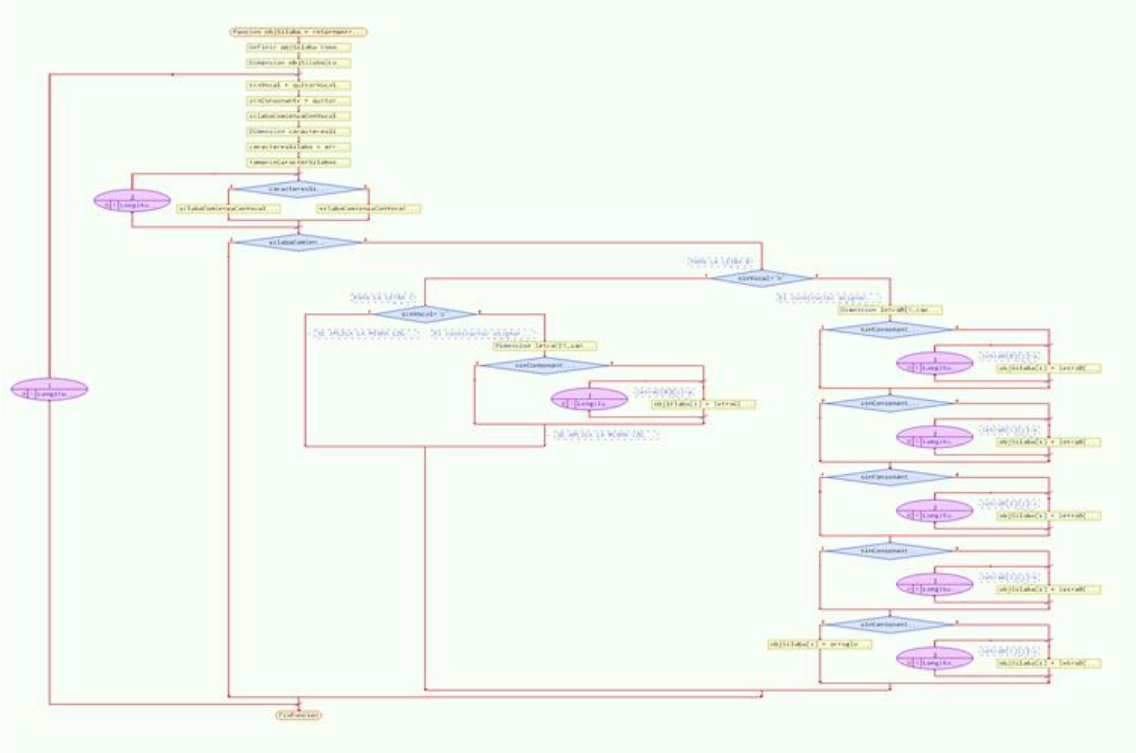


Figura 8 Diagrama de flujo Metaphone

Anexo 8: Estructura de combinación de fonemas

En la figura 9 se presenta la estructura en que se genera los fonemas de una palabra, por ejemplo, la palabra “bazuca”, sus posibles incorrecciones se van generando finalizando como se muestra en la estructura. *Ver figura 9*

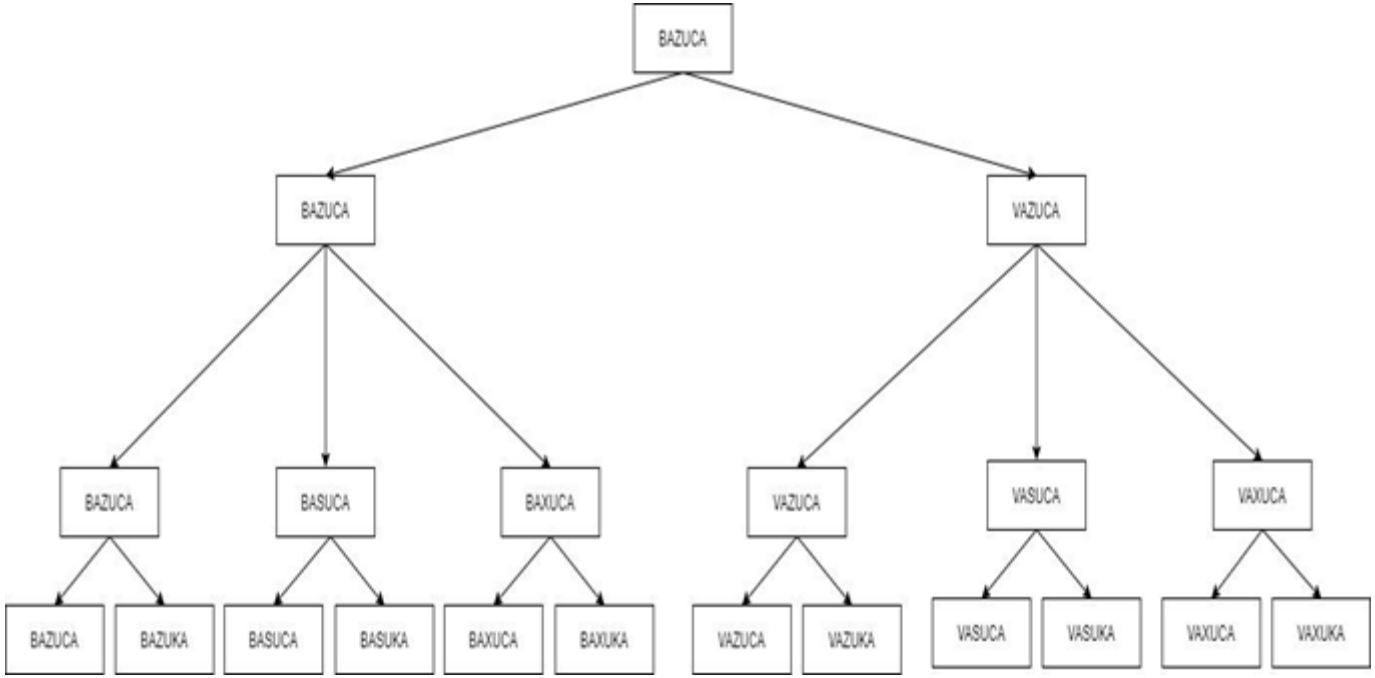


Figura 9 Estructura de combinación de fonemas

Anexo 9: Proceso de algoritmo Distancia de Levenshtein

Cada palabra obtenida de la base de datos en relación a su fonema, será comparada con la palabra incorrecta, para este proceso se llamará al método "almacenarMatriz()" de la Distancia de Levenshtein, enviando como parámetros el tamaño de la palabra correcta e incorrecta, y los caracteres de las mismas.

Inicializamos la Matriz de Levenshtein con las dimensiones respecto al tamaño de la palabra incorrecto y la palabra correcta. Asignamos valores en X del 0 hasta la longitud de la palabra incorrecta, y valores en Y del 0 hasta la longitud de la palabra correcta.

Se asigna valores a la matriz excepto en las columnas iniciales, cumpliendo las condiciones siguientes: Siendo "x" y "y" el índice de los caracteres de la palabra incorrecta e incorrecta respectivamente, si el carácterA_{x-1} es igual a caracterB_{y-1} entonces se asigna a la Matriz $_{[\,i,\,j]}$ con el valor mínimo entre Matriz $_{[\,i\,-\,1,\,j\,-\,1]}$, Matriz $_{[\,i\,-\,1,\,j]}$ y Matriz $_{[\,i,\,j\,-\,1]}$, de lo contrario, se asigna a la Matriz $_{[\,i,\,j]}$ el mismo valor mínimo más una unidad. *Ver figura 10*

```
Funcion numeroMenor←minimo ( entero1, entero2 )
    numeroMenor←0
    Si entero1<entero2 Entonces
        numeroMenor←entero1
    SiNo
        numeroMenor←entero2
    Fin Si
Fin Funcion

Funcion matriz←almacenarMatriz ( longitudA, longitudB ,caracteresA, caracteresB)
    Dimension matriz[longitudA, longitudB];
    Para i←0 Hasta longitudA Con Paso 1 Hacer
        matriz[i, 0]←i;
    Fin Para

    Para i←0 Hasta longitudB Con Paso 1 Hacer
        matriz[0, i]←i;
    Fin Para

    Para i←1 Hasta longitudA Con Paso 1 Hacer
        Para j←1 Hasta longitudB Con Paso 1 Hacer
            Si caracteresA[i-1] == caracteresB[j-1] Entonces
                matriz[i, j]←minimo(matriz[i-1,j-1],minimo(matriz[i-1,j], matriz[i,j-1]));
            SiNo
                matriz[i, j]←minimo(matriz[i-1,j-1],minimo(matriz[i-1,j], matriz[i,j-1]))+1;
            Fin Si
        Fin Para
    Fin Para
Fin Funcion

Algoritmo LevenshteinDistance
    Definir tamañoCaracterA Como Entero
    Definir tamañoCaracterB Como Entero

    Dimension caracteresA[tamañoCaracterA]
    Dimension caracteresB[tamañoCaracterB]

    Dimension matriz[tamañoCaracterA,tamañoCaracterB]
    matriz←almacenarMatriz(tamañoCaracterA, tamañoCaracterB, caracteresA, caracteresB); //DATO FINAL
FinAlgoritmo
```

Figura 10 Pseudocódigo de Distancia de Levenshtein

Posteriormente se muestra el diagrama de flujo de Distancia de Levenshtein el cual retorna la distancia más corta entre una palabra incorrecta a incorrecta. *Ver Figura 11*

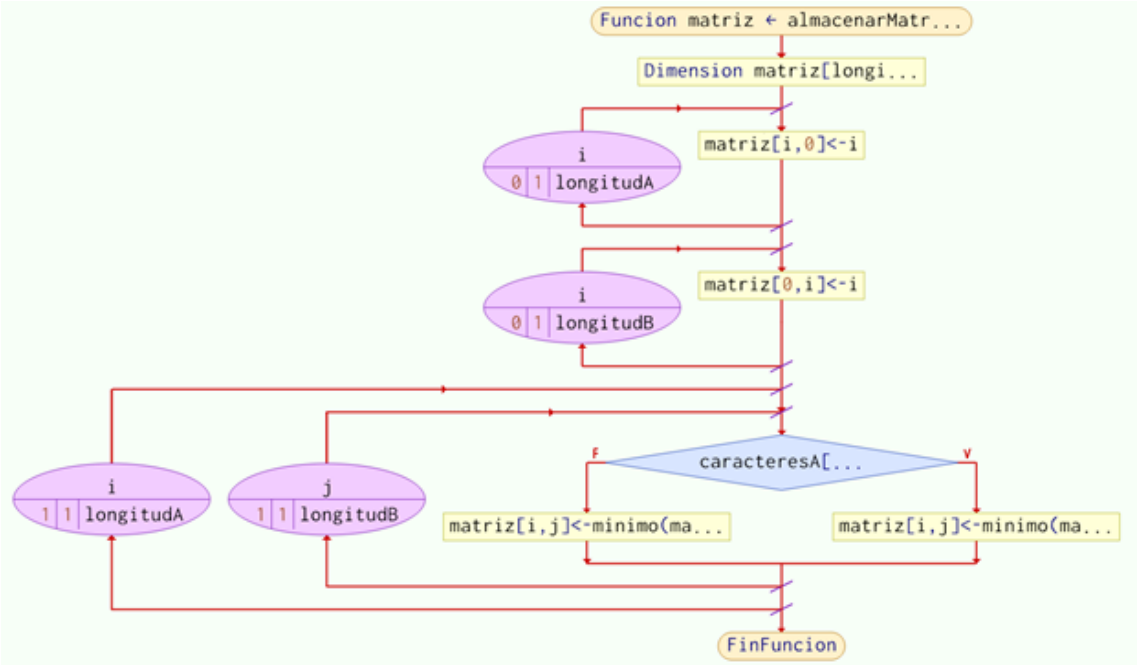


Figura 11 Diagrama de flujo de Distancia de Levenshtein – Retorna Distancia

Seguidamente se muestra el diagrama de flujo el cuál retorna la distancia menor entre todas las distancias encontradas en comparación con las palabras consultadas. *Ver Figura 12*

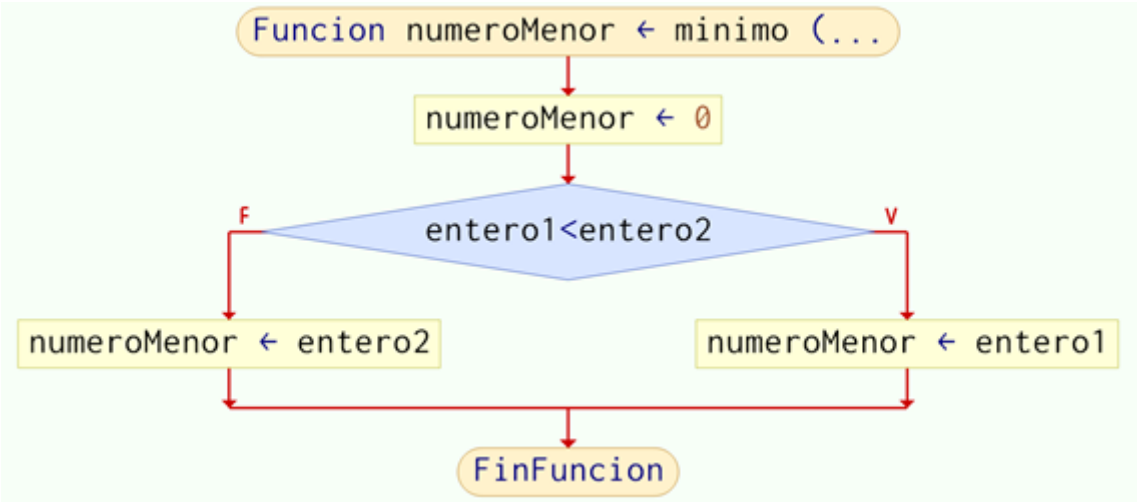


Figura 12 Diagrama de flujo de Distancia de Levenshtein – Retorna Distancia Menor

Anexo 10: Proceso del algoritmo MC Corrector

Se toma como parámetro principal el párrafo corregido ortográficamente. Este separa en un vector que llamaremos de momento “arreglo” clasificando las palabras, signos de puntuación, caracteres especiales y los conectores lógicos.

El arreglo será recorrido por el algoritmo, en el que cada índice será consultado en la base de datos para verificar si es una palabra homónima o no. Sí esto es falso, no se hará nada, de lo contrario, es posible que sea un error contextual. Por ejemplo, la palabra “votaré” es una palabra homonimia, ya que el usuario puede referiste a la palabra del verbo “votar” o del verbo “botar”.

Si la consulta a la tabla homonimia devuelve algún valor, se guardan tres datos relevantes, el nombre de la columna en la que fue encontrada la palabra, las palabras relacionadas y su id relación con otras palabras homónima. Cada una de estas palabras relacionadas serán comparadas con cada una de las palabras del párrafo, dándose que sí la igualdad es verdadera, se adicionará una unidad entera al contador en el que llamaremos $contador_a$, de lo contrario, seguirá recorriendo las palabras del párrafo en búsqueda de coincidencia de palabras relacionadas.

Se genera la cantidad de contadores del número de cantidad de id relación de las palabras homónimas. Por ejemplo, el verbo “basar” tiene tres id relación, con la palabra “bazar”, su conjugación verbal “basó” con “vaso” y también “bazo”. Teniendo así un total de tres relaciones por lo que se generaría tres contadores en total que llamaremos $contador_b$, $contador_c$ y $contador_d$.

Se repite el mismo proceso de asignación de valor de la variable $contador_a$ para la variable $contador_b$, $contador_c$ y $contador_d$, el contará las veces que ha encontrado similitud cada palabra relacionada con las palabras del párrafo.

Por último, se identificará el mayor de las variables: $contador_a$, $contador_b$, $contador_c$ y $contador_d$. Así se da pasó a la nueva palabra contextualizada, en el que haya tenido mayor cantidad de palabras relacionadas en el párrafo, se tomará como corrección en base al contexto y se reemplaza en el párrafo original.

Más adelante, en la figura 13 se muestra el pseudocódigo explicado anteriormente donde se resalta la secuencia algorítmica de variables, bucles, condiciones. *Ver Figura 13*

```
Algoritmo CorregirContexto
    parrafo←parametroParrafo //Se asignará el valor del parrafo

    Definir caracteresEspeciales Como Caracter
    Dimension caracteresEspeciales[cantCaracter]
    caracteresEspeciales[0]="("; caracteresEspeciales[7]=")"; caracteresEspeciales[8]="¿"; caracteresEspeciales[12]="?";
    caracteresEspeciales[1]="<"; caracteresEspeciales[6]=">"; caracteresEspeciales[9]="="; caracteresEspeciales[13]="*";
    caracteresEspeciales[2]="%"; caracteresEspeciales[5]="["; caracteresEspeciales[10]="]"; caracteresEspeciales[14]="@";
    caracteresEspeciales[3]="&"; caracteresEspeciales[4]=","; caracteresEspeciales[11]="."; caracteresEspeciales[15]=":";

    Dimension palabras[cantPalabras]

    Para i←0 Hasta Longitud(palabras[i]) Con Paso 1 Hacer
        palabras[i]=sustraerPalabras(parrafo)
    Fin Para

    Para i←0 Hasta Longitud(palabras[i]) Con Paso 1 Hacer
        palabras[i]=limpiarPalabras(palabras[i])
    Fin Para

    Definir contextualizado Como Caracter
    Definir  contadorHomonimia Como Entero

    Para i←0 Hasta Longitud(palabras[i]) Con Paso 1 Hacer
        data=consultarPalabrasRelacionada(palabras[i])
        Si data ≠ " " Entonces
            contadorHomonimia=contadorHomonimia+1
            posiblesPalabrasA=limpiarPalabras(data)
            contador=0
            Para j←0 Hasta Longitud(palabras[i]) Con Paso 1 Hacer
                Para k←0 Hasta Longitud(posiblesPalabrasA) Con Paso 1 Hacer
                    Si palabras[i]=posiblesPalabrasA Entonces
                        contador=contador+1
                    Fin Si
                Fin Para
            Fin Para
            idRelaciones=limpiarPalabras(data);
            Dimension contadorRelacionado[Longitud(idRelaciones)];
            Repetir
                m=0
                contadorRelacionado[m]=0
            Hasta Que m=Longitud(idRelaciones)

            indicadorMayor=contador
            Para j=0 Hasta Longitud(idRelaciones) Con Paso 1 Hacer
                dat=verificaRelaciones(data,idRelaciones)
                posiblesPalabrasB=limpiarPalabras(dat)

                Para k←0 Hasta Longitud(palabras[i]) Con Paso 1 Hacer
                    Para l←0 Hasta Longitud(posiblesPalabrasB) Con Paso 1 Hacer
                        Si palabras[i]=posiblesPalabrasB Entonces
                            contadorRelacionado[j]=contadorRelacionado[j]+1
                        Fin Si
                    Fin Para
                Fin Para

                Si indicadorMayor<contadorRelacionado[j] Entonces
                    indicadorMayor=contadorRelacionado[j]
                    palabras[i]=dat
                Fin Si
            Fin Para
        Fin Si
        contextualizado=+palabras[i]+" " //DATO FINAL
    Fin Para
```

Figura 13 Pseudocódigo de MC Corrector – Contexto

En la figura 14, se muestra el diagrama de flujo del algoritmo explicado en esta sección, donde indica gráficamente la secuencia del proceso de corrección contextual. *Ver Figura 14*

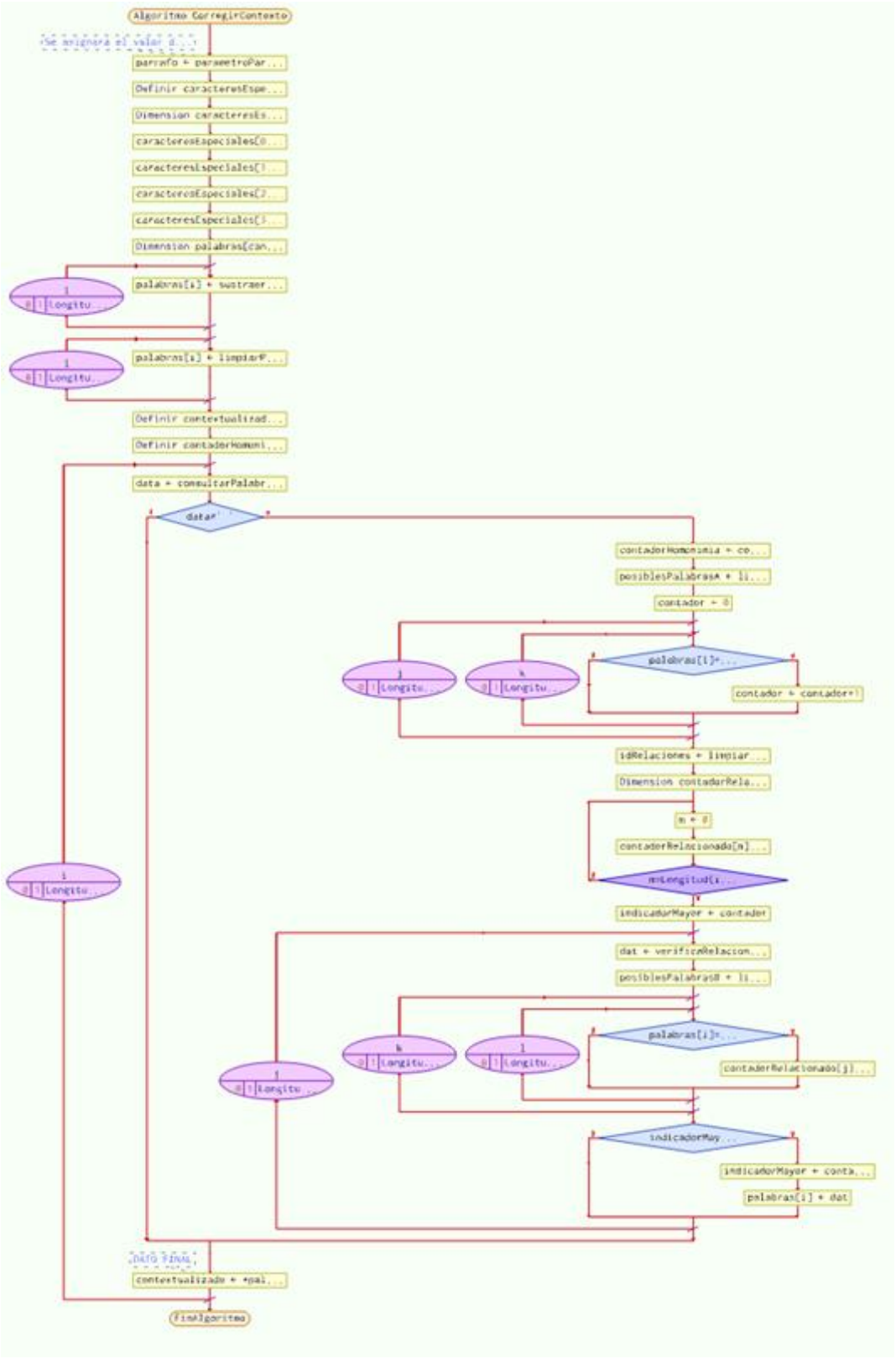


Figura 14 Diagrama de flujo MC Corrector - Contexto

Se obtiene la cantidad de caracteres del párrafo y la cantidad de caracteres del conector para verificar que este cumpla con la condición de que el párrafo debe ser mayor al conector. Es de importancia hacer esta verificación ya que en los pasos siguiente podría generarse un error lógico si no habría una validación previa.

Se recorre cada carácter del párrafo buscando una primera coincidencia para comenzar el proceso de conteo. Es de importancia definir una variable que te indique la cantidad restante, el cuál será el valor del tamaño de caracteres del párrafo menos el recorrido que tiene el bucle. Esto servirá para evitar otro error lógico en el que la cantidad restante pueda ser menor al tamaño de caracteres del conector a reemplazar.

Se recorre cada carácter del conector y se verifica el índice actual del párrafo en aumento con el índice del carácter actual en aumento. Sí estos coinciden, un contador de coincidencia aumenta una unidad así también como el índice actual del párrafo.

Sí el contador de coincidencia es igual al tamaño de los caracteres del conector, significa que coincidieron en todos sus caracteres, por ende, se adiciona una variable que indique la cantidad de repeticiones de coincidencia. Inicialmente, hay una coincidencia de conector lógico en el párrafo.

Sí el contador de coincidencia es mayor a 1, se declara una variable de valor aleatorio dentro del rango del tamaño de conectores equivalente para elegir un conector lógico aleatorio.

Sí el valor aleatorio es diferente del indicador del bucle inicial, o sea, si el número aleatorio no coincide con el índice del conector lógico, se reemplaza el conector lógico del párrafo original por el propuesto en número aleatorio, por último, adicionando al indicador del bucle inicial el tamaño del conector. Aquí se toma en cuenta la validación previa del paso (a), ya que el párrafo variará en menor o mayor tamaño dependiente del tamaño del conector a reemplazar.

Seguidamente se muestra el pseudocódigo en el cual se muestra el proceso secuencial algorítmico, así también se visualiza las variables y sus tipos, como los bucles y condiciones. *Ver Figura 15*

```
Funcion parrafoCorregido ← redundanciaCorregido ( parrafo , conector)
    Definir parrafoCorregido Como caracter
    Definir contieneConectores Como Logico
    contieneConectores←Falso //parrafo.contains(conector[i]) consulta si existe el conector en el parrafo

    Si contieneConectores Entonces
        Para i←0 Hasta Longitud(conector) Con Paso 1 Hacer
            tamConector←Longitud(conector[i])
            tamParrafo←Longitud(parrafo)
            repeticionEncontrada←0
            Si tamParrafo ≥ tamConector Entonces
                Definir caracterParrafo Como Caracter
                Definir caracterConector Como Caracter
                Dimension caracterParrafo[tamParrafo]
                Dimension caracterConector[tamConector]

                Para j←0 Hasta Longitud(caracterParrafo[i]) Con Paso 1 Hacer
                    contadorCoincidencia←0
                    aumentCaracter←j
                    cantRestante←Longitud(caracterParrafo[i])-(j+1)
                    Si cantRestante ≥ Longitud(caracterConector[j]) Entonces
                        Si caracterParrafo[j]=caracterConector[0] Entonces
                            Para k←0 Hasta Longitud(caracterConector[j]) Con Paso 1 Hacer
                                Si caracterParrafo[aumentCaracter]=caracterConector[k] Entonces
                                    contadorCoincidencia=+1
                                    aumentCaracter=+1
                                Fin Si
                            Fin Para
                        Fin Si
                        Si contadorCoincidencia=Longitud(conector) Entonces
                            repeticionEncontrada=+1
                            Si repeticionEncontrada>1 Entonces
                                Definir aleatoriedad, salir Como Entero
                                Repetir
                                    aleatoriedad←AleatorioConector //random.nextInt(conector.length); Devuelve un numero aleatorio para obt
                                    Si aleatoriedad≠i Entonces
                                        salir←1
                                        parrafoCorregido←conector[aleatoriedad]
                                        j=+Longitud(conector[i])
                                    Fin Si
                                Hasta Que salir=1
                            SiNo
                                parrafoCorregido=+caracterParrafo[j]
                            Fin Si
                        SiNo
                            parrafoCorregido=+caracterParrafo[j]
                        Fin Si

                    Fin Para
                Fin Si
            Fin Para
        SiNo
            parrafoCorregido←parrafo
        Fin Si
Fin Funcion
```

Figura 15 Pseudocódigo de MC Corrector – Redundancia

En la figura 16 se muestra el diagrama de flujo del algoritmo explicado en esta sección, explica gráficamente la secuencia del proceso de corrección contextual. *Ver Figura 16*

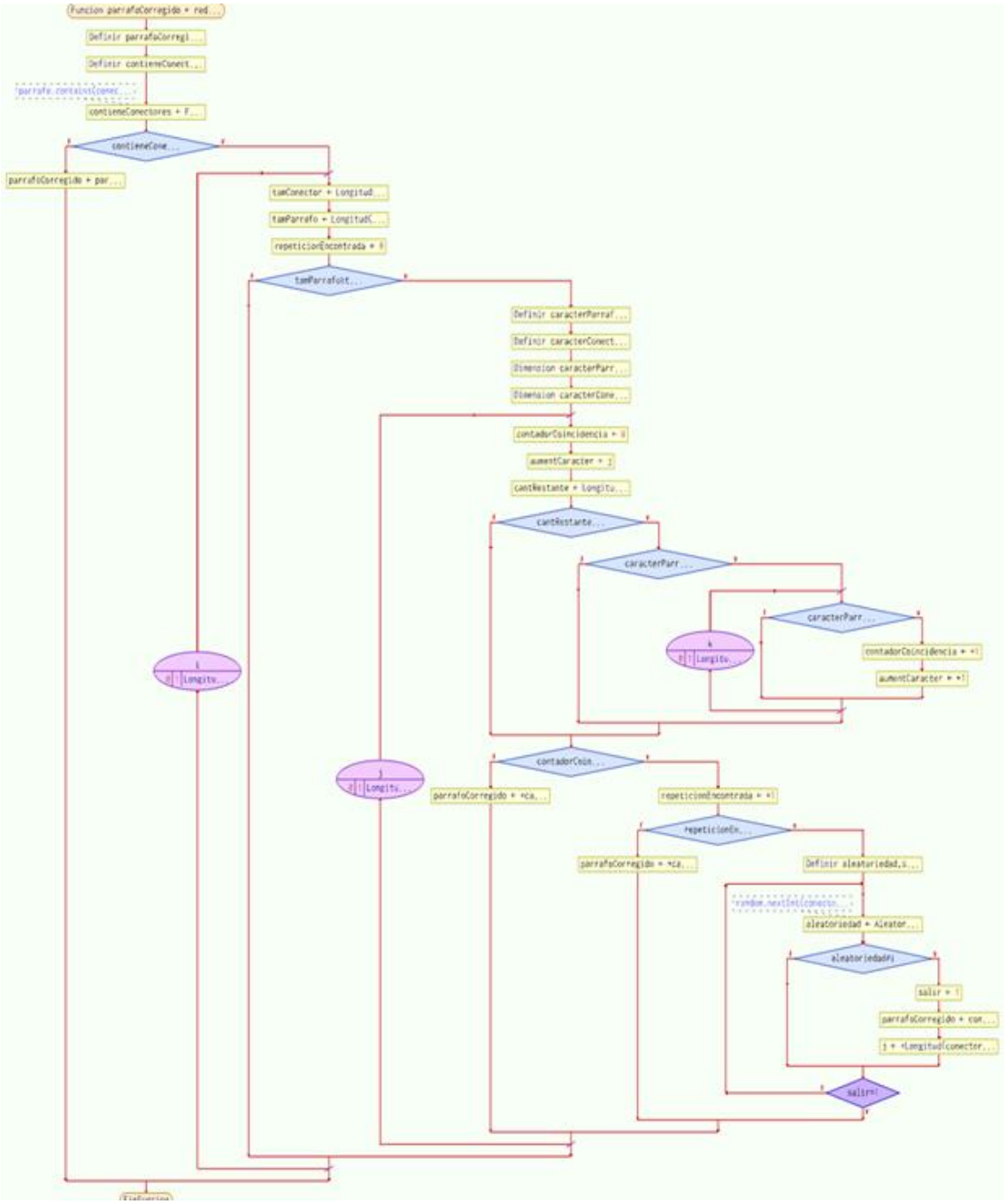


Figura 16 Diagrama de flujo MC Corrector - Redundancia

Anexo 11: Prototipo

En la figura 17 se observa la pantalla principal donde se ingresará el texto con errores. *Ver Figura 17*

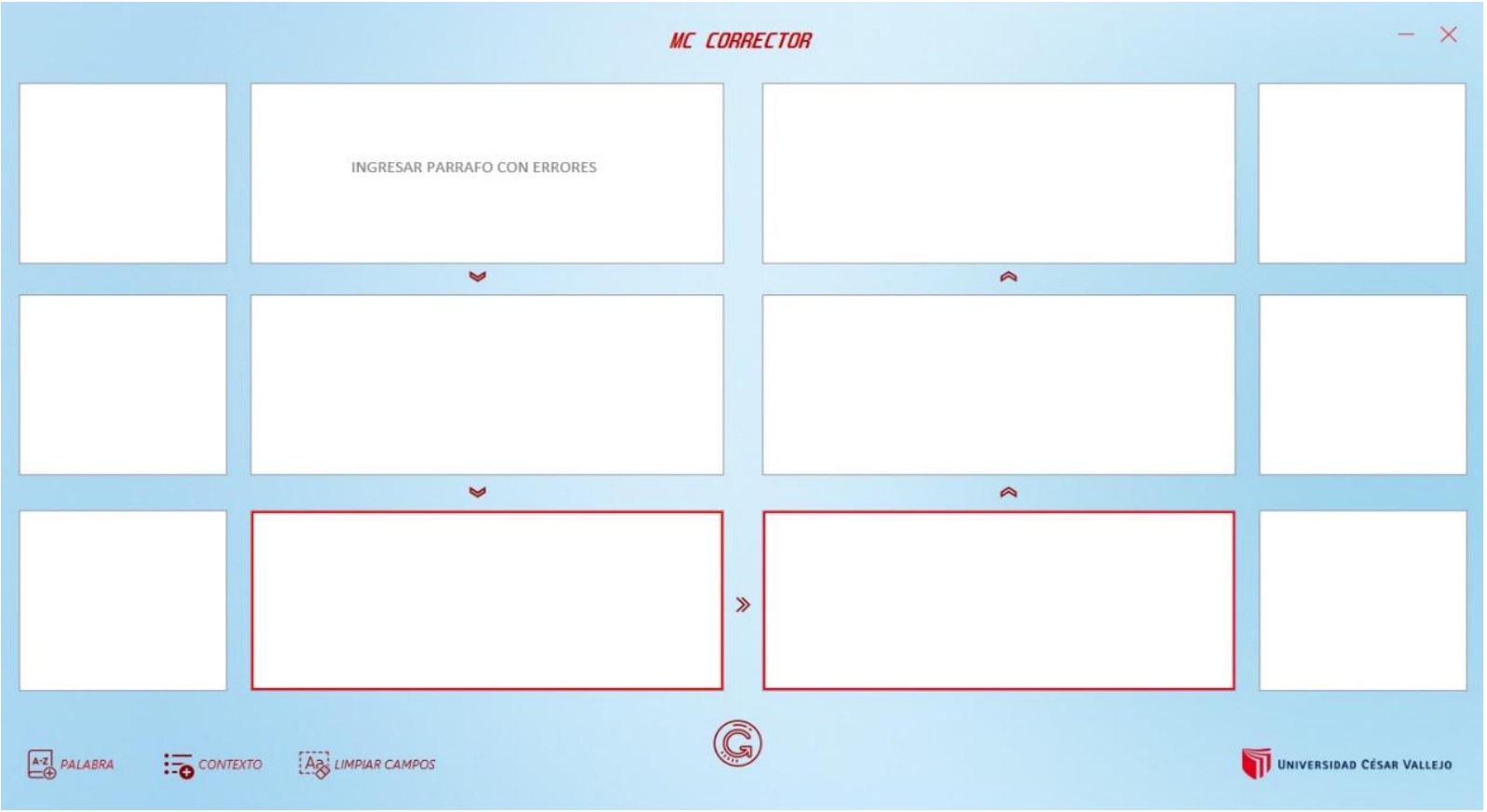


Figura 17 Interfaz principal

Anexo 12: Manual de usuario de la herramienta desarrollada

Para visualizar los datos se debe tener en cuenta el orden de las cajas de texto. Se visualiza en el siguiente orden:

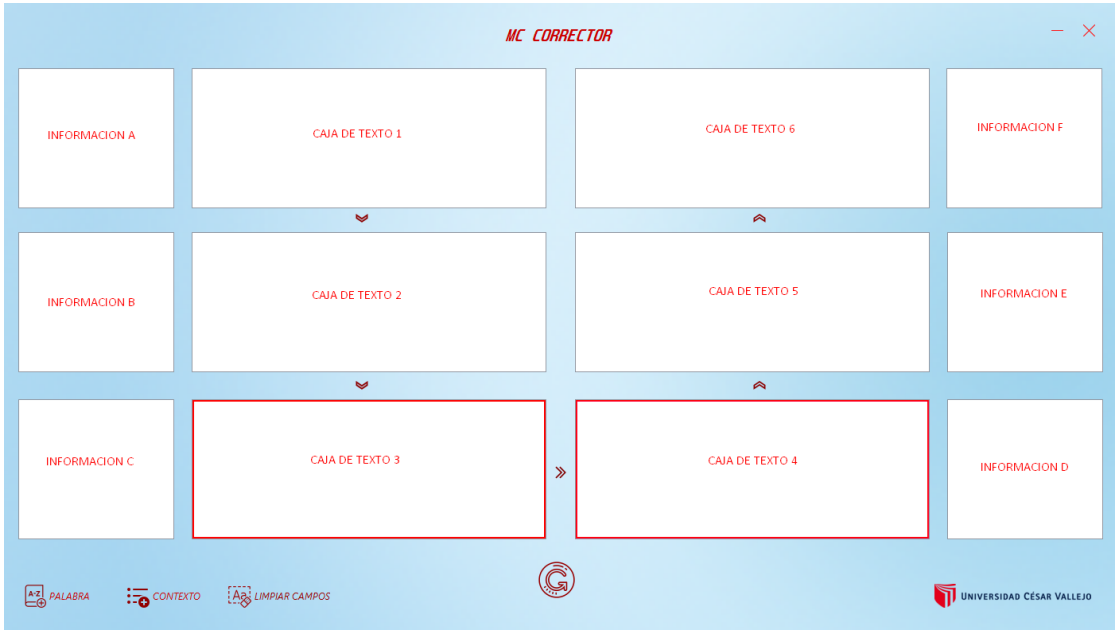


Figura 18 Secuencia de las caja de texto

Caja de texto 1: Se ingresará el párrafo inicial con errores ortográficos, gramaticales y de estilo, y al lado izquierdo se muestra la información (A) donde indica la cantidad de palabras, caracteres y palabras cortas. A continuación, hacemos clic en el botón circular central.

Figura 19 Botón Corregir Párrafo

Caja de texto 2: Mostrará el párrafo con los errores ortográficos corregidos, y a su derecha muestra la información (B) donde indica la cantidad de errores ortográficos, las palabras incorrectas con sus respectivos fonemas y las palabras posibles a corregir.

Caja de texto 3: Mostrará el párrafo con los errores gramaticales corregidos, puntualmente las palabras con errores contextuales, y a su derecha se muestra la información (C) donde se indica la cantidad de errores gramaticales, la palabra incorrecta, las palabras en base a su contexto, y la cantidad de palabras relacionadas.

Caja de texto 4: Mostrará el párrafo con los errores de estilos corregidos, puntualmente los errores de conectores lógicos redundantes, y a su izquierda se muestra la información (D) donde se indica la cantidad de redundancia y el reemplazo aleatorio de un conector equivalente.

Caja de texto 5: Mostrará el párrafo con los errores gramaticales corregidos, puntualmente los errores copulativos de y/e - o/u, y a su izquierda se muestra la información (E) donde se indica la cantidad de errores copulativos, el error y su corrección.

Caja de texto 6: Mostrará el párrafo final corregido con las correcciones adicionales de mayúsculas y espacios innecesarios, y a su izquierda se muestra la información (f) donde indica las palabras modificadas.

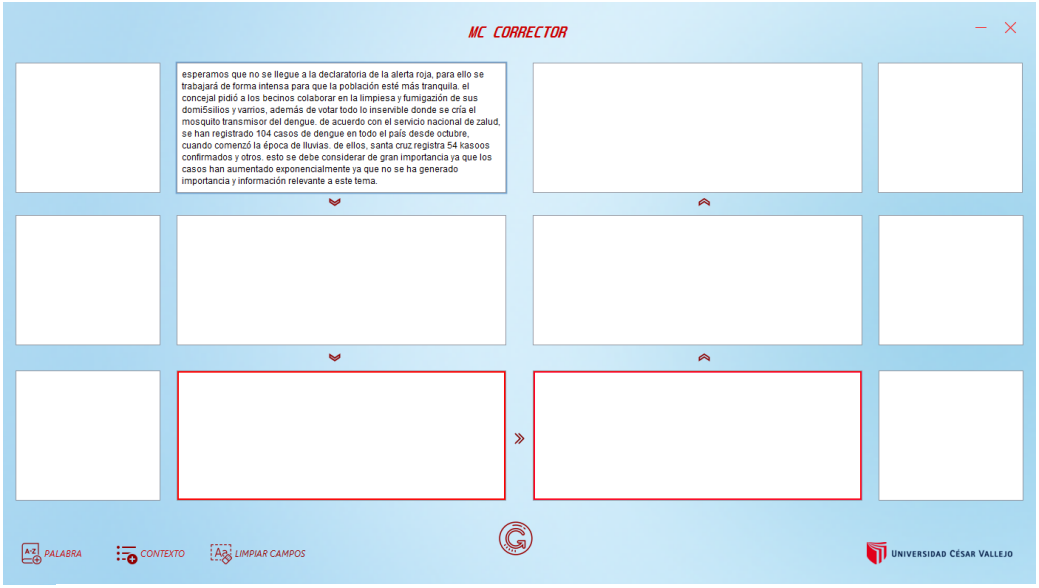


Figura 20 Ingreso de texto con errores a corregir

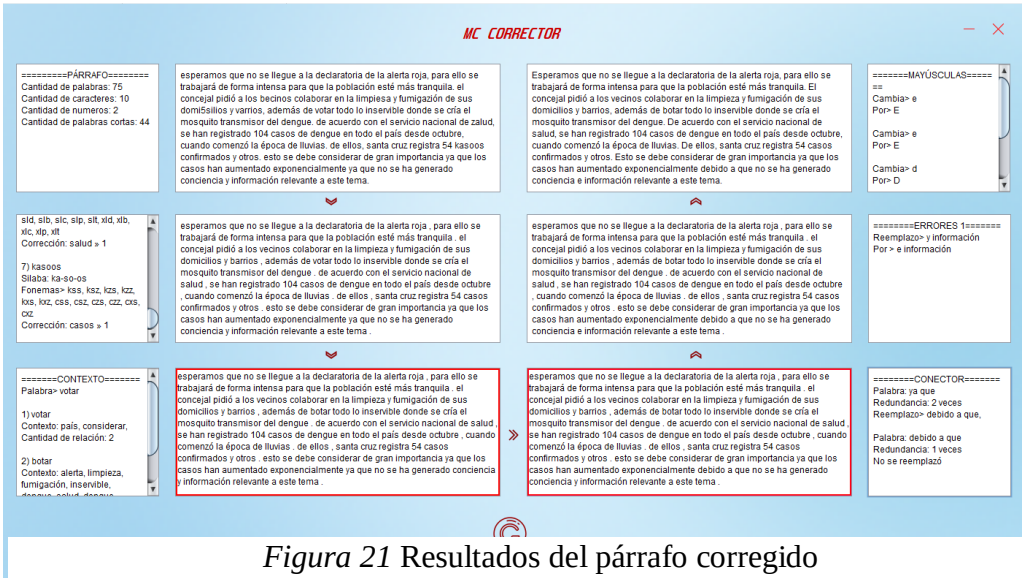


Figura 21 Resultados del párrafo corregido

Botón PALABRA: Este botón se usará para la gestión de la base de datos. Guarda una palabra en la base de datos “dic_espa”. En caso de que el programa detecte una palabra correcta como incorrecta, se deberá utilizar este botón para almacenar, así en una siguiente consulta donde contenga esa palabra podrá detectarlo y corregirlo adecuadamente si está mal escrita, o no corregirla si está correcta.

Paso 1: Se debe subrayar la palabra detectada en la caja de texto información(b) y seleccionar el botón “Palabra”.

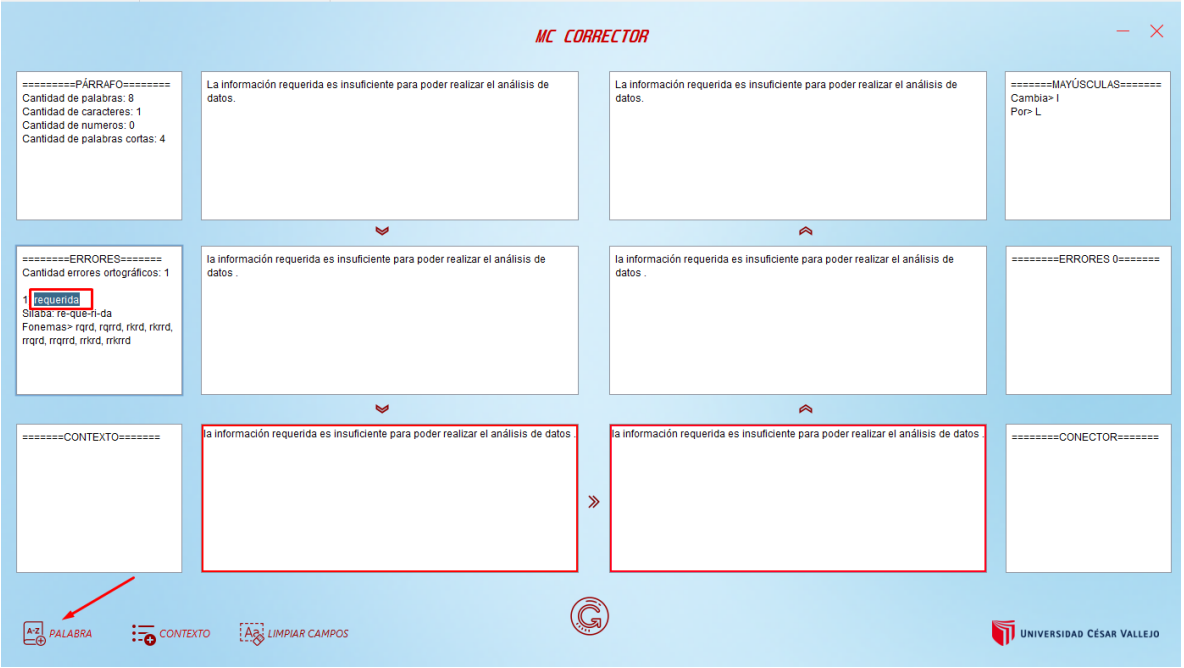


Figura 22 Seleccionar palabra

Paso 2**:** En una siguiente consulta podrá corregir el error en la palabra.

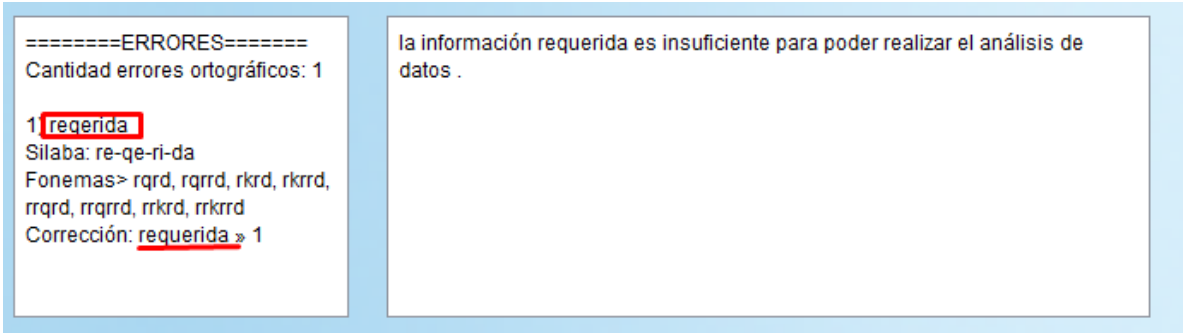


Figura 23 Corrección ortográfica

Botón CONTEXTO: Este botón se usará para la gestión de la base de datos, es más para uso administrativo en caso se requiera mejorar la precisión de la detección de palabras relacionadas en base al contexto.

Paso 1: Seleccionar la palabra en la caja de texto 1. Se debe seleccionar la palabra completa y debe ser una homonimia, y luego hacer clic en el botón “Contexto”.

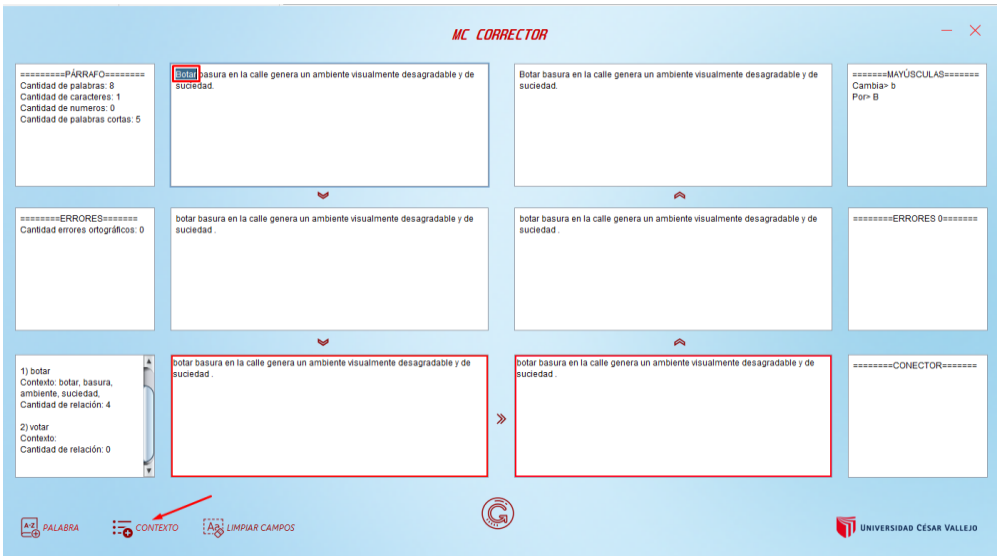


Figura 24 Guardar palabra relacionada – Parte 1

Paso 2: Ingresar la palabra relacionada contextualmente a la palabra homónima y clic en “Aceptar”.

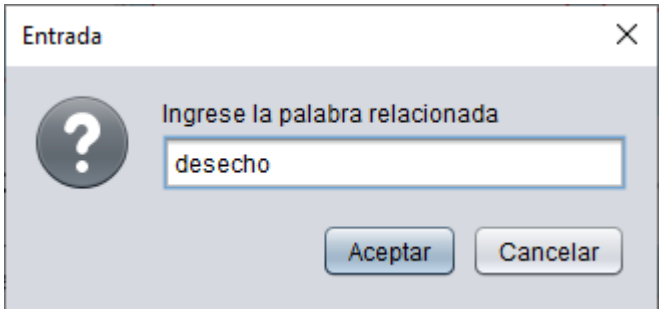


Figura 25 Guardar palabra relacionada – Parte *2*

Botón LIMPIAR: Este botón limpia todas las cajas de texto para poder iniciar una nueva corrección.

Figura 26 Limpiar campos

Anexo 13: Lista de palabras Homófonas

Las palabras homófonas son palabras que se suenan absolutamente igual que otra sin embargo su escritura y significado es diferente. A continuación, se muestra la tabla con palabras homófonas, categoría y significado de cada una.

Tabla 10 *Palabras homófonas*

Palabra	Categoría	Significado
bobina	sust	Cilindro de hilo, cordel, etc., arrollado en torno a un canuto de cartón u otra materia.
bovina	adj	Perteneciente o relativo al toro o a la vaca.
cien	adj, pron	Adjetivo pronominal numeral.
sien	sust	Cada una de las dos partes laterales de la cabeza situadas entre la frente, la oreja y la mejilla.
graba	verb	Forma del verbo grabar.
grava	sust	Piedra, material de construcción.
grabado	sust, verb	Arte de grabar.
gravado	verb	Forma del verbo gravar.
horca	sust	Conjunto de uno o dos palos verticales sujetos al suelo y otro horizontal del cual se cuelga por el cuello, para dar muerte a los condenados a esta pena.
orca	sust	Cetáceo.
cause	verb	Del verbo causar.
cauce	sust	Lecho de un río.
ablando	verb	Del verbo ablandar
hablando	verb	Del verbo hablar
abollar	verb	Producir una depresión en una superficie con un golpe o apretándola.
aboyar	verb	Poner boyas.
alaba	verb	Elogiar, celebrar con palabras.
halaba	verb	Forma del verbo halar.
aremos	verb	Remover la tierra haciendo surcos con el arado.
haremos	verb	Del verbo hacer.
baqueta	sust	Palillos con que se toca el tambor.

vaqueta	sust	Cuero de ternera, curtido y adobado.
barón	sust	Título de dignidad.
varón	sust	Ser humano de sexo masculino.
bate	sust	Palo con el que se golpea la pelota.
vate	sust	Adivino, poeta.
bota	sust, verb	Calzado. //Del verbo botar.
vota	verb	Del verbo votar.
botar	verb	Arrojar, desechar.
votar	verb	Dar el voto, elegir.
cabe	verb	Del verbo caber, entrar.
cave	verb	Del verbo cavar.
calló	verb	Del verbo callar.
cayó	verb	Del verbo caer.
casa	sust, verb	Edificio para habitar// Del verbo casar.
caza	sust, verb	Acción de cazar// del verbo cazar.
casar	verb	Contraer matrimonio.
cazar	verb	Perseguir animales para matarlos.
cegar	verb	Quitar la vista a alguien.
segar	verb	Cortar mieses.
cerrar	verb	Asegurar con cerradura.
serrar	verb	Cortar o dividir con la sierra.
ciervo	sust	Animal mamífero rumiante.
siervo	sust	Esclavo.
cocer	verb	Hacer comestible un alimento crudo sometiéndolo a ebullición o a la acción del vapor.
coser	verb	Unir con hilo, generalmente enhebrado en la aguja, dos o más pedazos de tela, cuero u otra materia.
cocido	verb, adj	Del verbo cocer/cocinar.
cosido	verb, adj	Del verbo coser.
errar	verb	No acertar.
herrar	verb	Ajustar y clavar las herraduras a las caballerías, o los callos a los bueyes
espiar	verb	Observar secretamente.

expiar	verb	Pagar [por una culpa o delito].
espirar	verb	Expulsar el aire por los pulmones.
expirar	verb	Morir.
hice / hizo	verb	Forma del verbo hacer.
ice / izo	verb	Forma del verbo izar.
hojear	verb	Mover o pasar ligeramente las hojas de un libro o de un cuaderno.
ojear	verb	Mirar a alguna parte.
losa	sust	Piedra llana y de poco grueso, casi siempre labrada, que sirve para solar y otros usos.
loza	sust	Barro fino, cocido y barnizado, de que están hechos platos, tazas.
rallar	verb	Desmenuzar algo restregándolo con el rallador.
rayar	verb	Hacer o tirar rayas.
rallo	sust, Verb	Forma del verbo rallar. Utensilio de rallar.
rayo	sust, verb	Línea de luz que procede de un cuerpo luminoso, y especialmente las que vienen del Sol.
rebosar	verb	Dicho de una materia líquida: Derramarse por encima de los bordes del recipiente que la contiene.
rebozar	verb	Bañar un alimento en huevo batido, harina, miel, etc.
caso	sust, verb	Suceso, acontecimiento. //Del verbo casar.
cazo	sust, verb	Recipiente de cocina, de metal, porcelana, etc.// Del verbo cazar.
asía	verb	Del verbo asir.
hacía	verb	Del verbo hacer.
sexo	sust	Condición orgánica.
seso	sust	Cerebro, sentido.
acerbo	adj	Que es áspero en el sabor y en el olor.
acervo	sust	Conjunto de bienes o valores morales o culturales.
agito	verb	Mover una cosa rápidamente de un lado y otro.
ajito	sust	Bulbo de esta planta que se emplea como condimento.
ala	sust	Miembro de algunas aves e insecto que le sirven para volar.

hala	verb	Del verbo halar.
arrollo	verb	del verbo arrollar (atropellar)
arroyo	sust	Corriente pequeña de agua.
arte	sust	Obra o actividad en la que se muestra con ingenio un aspecto de la realidad.
harte	verb	Saciar el apetito de comer o beber.
asar	verb	Hacer comestible un alimento por la acción directa del fuego, o la del aire caldeado, a veces rociando aquel con grasa o con algún líquido.
azar	sust	Casualidad, caso fortuito.
asta	sust	Palo o barra en la que se coloca la bandera.
hasta	prep	Preposición.
aya	sust	Persona encargada en las casas principales de custodiar niños o jóvenes y de cuidar de su crianza y educación.
halla	verb	Dar con alguien o con algo que se busca.
haya	sust, verb	Árbol de la familia de las Fagáceas. // Del Verbo haber.
bacilo	sust	Bacteria en forma de bastoncillo.
vacilo	verb	Dicho de una cosa, moverse, titubear, tomar el pelo.
bares	sust	Plural de bar.
vares	verb	Del verbo varar.
basar	verb	Asentar algo sobre una base.
bazar	sust	Tienda en que se venden productos de varias industrias, comúnmente a precio fijo.
baso	verb	Del verbo basar.
vaso	sust	Pieza cóncava para contener alguna cosa.
basta	sust, interj	Cada una de las puntadas o ataduras que suele tener a trechos el colchón de lana para mantener esta en su lugar// Interjección.
vasta	adj	Dilatado, muy extendido o grande.
basto	Sust, verb	Cada uno de los naipes de este palo en la baraja. // Del verbo bastar
vasto	adj	Dilatado, muy extendido o grande.
bello	adj	Que tiene belleza.

vello	sust	Pelo que sale en algunas partes del cuerpo.
beses	verb	Forma del verbo besar.
veces	sust	Plural de vez.
cabo	sust	Accidente geográfico. // Mando militar del ejército. // Extremo de una punta.
cavo	verb	Del verbo cavar.
callado	adj, verb	Reservado, silenciado// parte del verbo callar.
cayado	sust	Palo o bastón corvo por la parte superior, especialmente el de los pastores para prender y retener las reses.
callo	sust, verb	Dureza que, por presión, roce y a veces lesión se forma en tejidos animales o vegetales//Del verbo callar.
cayo	sust	Cada una de las islas rasas, arenosas, frecuentemente anegadizas y cubiertas en gran parte de mangle, muy comunes en el mar de las Antillas y en el golfo mexicano.
caso	sust, verb	Suceso, acontecimiento. //Del verbo casar.
cazo	sust, verb	Recipiente de cocina, de metal, porcelana, etc., // Del verbo cazar.
cebo	sust, verb	Comida que se da a los animales para alimentarlos, engordarlos o atraerlos.
sebo	sust	Grasa sólida y dura que se saca de los animales herbívoros, y que, derretida, sirve para hacer velas, jabones y para otros usos.
ceda	verb	Del verbo ceder
seda	sust, verb	tela, tipo de hilo// Del verbo sedar.
cede	verb	Del verbo ceder
sede	sust, Verb	Lugar donde tiene su domicilio una entidad económica, literaria, deportiva, etc.// Del verbo sedar.
cenado	verb (part)	Participio del verbo cenar.
senado	sust	Cuerpo colegislador formado por personas elegidas o designadas en virtud de su cualificación, cargo, título, etc.
ciego	sust, adj, verb	Privado de la vista. // Del verbo cegar.
siego	verb	Del verbo segar.

cierra	verb	Del verbo cerrar.
sierra	sust, verb	Herramienta// pez// Del verbo serrar.
cita	sust, verb	Reunión o encuentro entre dos o más personas, previamente acordado// Del verbo citar.
sita	adj	Situado o fundado.
consciente	adj	Que siente, piensa, quiere y obra con conocimiento de lo que hace.
consiente	verb	Forma del verbo consentir.
contesto	verb	del verbo contestar.
contexto	sust	Entorno físico o de situación.
desecho	sust, verb	Residuo, desperdicio o cosa que se desecha.
deshecho	adj, verb (part)	Participio del verbo deshacer.
echo	verb	Forma del verbo echar.
hecho	verb(part), sust, adj	Participio del verbo hacer.
encima	adv	Es una posición o parte superior.
enzima	sust	Proteína que cataliza específicamente cada una de las reacciones bioquímicas del metabolismo.
estirpe	sust	Raíz y tronco de una familia o linaje.
extirpe	verb	Forma del verbo extirpar.
grabe	verb	Forma del verbo grabar.
grave	adj	Enfermo de cuidado.
hierba	sust	Planta.
hierva	verb	Forma del verbo hervir.
hinca	verbo	Del verbo hincar.
inca	sust, adj	Perteneciente o relativo a los aborígenes, de la parte oeste de América del sur.
hoces	sust	Plural de hoz.
oses	verb	Del verbo osar.
hola	interj	Interjección.
ola	sust	Onda de gran amplitud que se forma en la superficie de las aguas.

honda	adj, sust	Que tiene profundidad.
onda	sust	Cada una de las elevaciones que se forman al perturbar la superficie de un líquido.
hora	sust	Tiempo que equivale a 60 minutos, es decir, 3600 segundos.
ora	verb, conj	Del verbo orar// Conjunción.
hulla	sust	Carbón de piedra que se conglutina al arder y, calcinado en vasos cerrados, da coque.
huya	verb	Del verbo huir.
huso	sust	Instrumento manual para hilar.
uso	verb, sust	Del verbo usar.
mallo	verb	Del verbo mallar.
mayo	sust	Mes.
malla	sust, verb	Vestido de tejido de punto muy fino que, ajustado al cuerpo, usan en sus actuaciones los artistas de circo, bailarinas, etc.\\Del verbo mallar.
maya	adj, sust	Se dice del individuo de cualquiera de las tribus indias que hoy habitan principalmente el Yucatán, Guatemala y otras regiones adyacentes.
masa	sust, verb	Magnitud física que expresa la cantidad de materia que contiene un cuerpo. Del verbo masar.
maza	sust	Arma antigua de palo guarnecido de hierro, o toda de hierro, con la cabeza gruesa.
ollera	sust	Fabricante o vendedor de ollas y otros utensilios de barro.
oyera	verb	Forma del verbo oír.
paces	sust	Hacer la paz
pases	sust, verb	Plural de pase//del verbo pasar.
poso	verb	Del verbo posar
pozo	sust	Perforación que se hace en la tierra para buscar una vena de agua.
risa	sust	Movimiento de la boca y otras partes del rostro, que demuestra alegría.
riza	verb, adj	Forma del verbo rizar.

sabia	sust, adj	Dicho de una persona que
		posee la sabiduría.
savia	sust	Líquido que circula por los vasos de las plantas pteridofitas y fanerógamas y del cual toman las células las sustancias que necesitan para su nutrición.
sumo	adj, verb, sust	Forma del verbo sumar. // Arte marcial de origen japonés. //Máximo.
zumo	sust	Líquido de las hierbas, flores, frutas u otras cosas semejantes, que se saca exprimiéndolas o majándolas.
tasa	sust, verb	Relación entre dos magnitudes. // Del verbo tasar
taza	sust	Vasija pequeña para tomar líquidos.
testo	verb	Del verbo testar.
texto	sust	Todo lo que se dice en el cuerpo de un documento u obra manuscrita o impresa.
tubo	sust	Pieza hueca, de forma por lo común cilíndrica y generalmente abierta por ambos extremos.
tuvo	verb	Del verbo tener.
verás	verb	Del verbo ver.
veraz	adj	Que dice, usa o profesa siempre la verdad.
ves	verb	Del verbo ver.
vez	sust	Ocasión.
baya	sust, adj	Tipo de fruto carnoso con semillas rodeadas de pulpa.
valla	sust	Línea o término formado de estacas hincadas en el suelo o de tablas unidas, para cerrar algún sitio o señalarlo.
vaya	verb, interj, sust	del verbo ir// interj//Burla o mofa que se hace de uno o chasco que se le da.
bidente	adj	Que tiene dos dientes.
vidente	adj, sust	Que puede ver.
reciente	adj	Nuevo, fresco o acabado de hacer.
resiente	verb	Empezar a flaquear.
mesa	sust	Mueble.
meza	verb	Del verbo mecer.
seno	sust	Pecho.

ceno	verb	Del verbo cenar.
meses	sust	Plural de mes.
meces	verb	Del verbo mecer.
vota	verb	Del verbo votar.
bota	verb, sust	Del verbo botar//Calzado.
vote	verb	Del verbo votar
bote	verb, sust	Del verbo botar//Salto.
peces	sust	Plural de pez
peses	verb	Del verbo pesar.
reces	verb	Del verbo rezar.
reses	sust	Plural de res.
raza	sust	Casta o calidad del origen o linaje.
rasa	adj	Llana, lisa.
roza	verb	Del verbo rozar.
rosa	sust	Flor.
extirpe	verb	Del verbo extirpar, arrancar de cuajo.
estirpe	sust	Raíz, linaje.

Anexo 14: Lista de conectores lógicos textuales

Los conectores lógicos son palabras o expresiones que ayudan al escritor o lector relacionar ideas el cual ayuda a mejorar la coherencia de una oración o frase. Mediante estos conectores el escritor relaciona coherente y lógicamente una idea con otra. Por otro lado, el lector se deja influir por los conectores para entender la conexión que hay entre una nueva información con la distribuida anteriormente.

Tabla 11 *Conectores lógicos textuales*

Conectores	
Adición	Además, incluso, inclusive, asimismo, igualmente, de igual manera, modo, forma, es más, del mismo modo, de la misma manera, forma, encima, por añadidura
Consecuencia	Por lo que, por ello, eso, así que, entonces, de manera, modo que, de ahí que, de ello resulta que, por ese, tal, dicho motivo, razón, causa, por lo tanto, en consecuencia, por consiguiente, por ende, de donde se sigue, de suerte que, en efecto, así pues, pues, luego
Contra argumentación	Pero, mientras que, sin embargo, a pesar de, pese a, sino, no obstante, no obstante, en lugar, vez de , con todo, aun así, si bien, en tanto que, después de todo, el contrario, sea como sea, en cambio , ahora bien , antes bien , contrariamente , excepto si, a no ser que, de todos modos, de todas maneras, formas, en cualquier caso, así y todo
Causal	Porque, ya que, pues, gracias a, por eso, a causa de ello, puesto que, dado que, por el hecho de que, visto que, supuesto que, como, en virtud de
Condicionales	Cuando, siempre y cuando, siempre que, mientras, según, a no ser que, sólo que, con tal que, a menos que, en el caso de que, con que
Finales	Para, por, con el fin de, con miras a, a fin de, con el propósito de, con el objeto de, con ánimo de, con vistas a, con idea de, con intención de, a efectos de, con motivo de, con pretexto de

Anexo 15: Reglas fonéticas del español

Los fonemas de una palabra es la conjugación de la unidad sonora de cada una de sus silabas, en el que se puede diferenciar el sonido de la totalidad de la palabra. En la tabla 9999 se muestra las reglas de los fonemas para el idioma español para el algoritmo Metaphone. Fonemas a tener en cuenta: B, CH, D, F, G, J, K, L, M, N, P, R, S, T, W, Y. Cabe resaltar que se han agregado algunas más, como por ejemplo “x” por la “s”, “z”.

Tabla 12 *Reglas fonéticas del español para el algoritmo Metaphone*

Letra	Transformación	Condición
C	S	Si es CE o CI.
	K	Si es CA, CO o CU.
G	J	Si es GE o GI
	G	Si es GA, GO o GU
LL	Y	En todas las ocasiones
Q	K	Si es QA, QUE o QUI
	C	Si es QA
RR	R	En todas las ocasiones
V	B	En todas las ocasiones
W	GU	En todas las ocasiones
X	S	Si al inicio de palabra
	KS	Si no es al inicio de palabra
Z	S	En todas las ocasiones
Y	I	Si no es seguida de una vocal
M	N	Si MB o MP
P	G	Si es PT
GN	N	En todas las ocasiones
PS	S	En todas las ocasiones
FT	T	En todas las ocasiones
MN	N	En todas las ocasiones
PT	T	En todas las ocasiones
TS	S	En todas las ocasiones