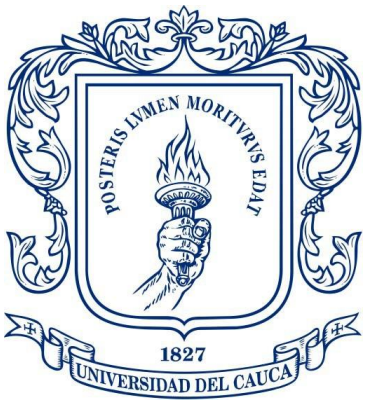
Prototipo software para la identificación de partes del discurso (Part-Of-Speech Tagging) utilizando un enfoque metaheurístico



Trabajo de Grado

Jose Julio Tobar Cifuentes

Miguel Alexis Solano Jiménez

Director:
Ph.D. Luz Marina Sierra Martínez

Codirector:
Ph.D. Carlos Alberto Cobos Lozada

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Línea de Investigación Sistemas Inteligentes
Ingeniería de Sistemas

Popayán, mayo de 2020

A nuestras familias

Agradecimientos

Al finalizar nuestra tesis de pregrado, queremos agradecer a todas las personas e instituciones que nos brindaron el apoyo para culminar esta tan anhelada meta.

Agradecemos a nuestros directores de tesis, Luz Marina Sierra Martínez y Carlos Alberto Cobos, por todo el apoyo para el desarrollo y ejecución de este proyecto. Nuestros sinceros agradecimientos.

Muchas gracias a todos nuestros compañeros y amigos que hicimos en nuestra carrera, también son parte de este reconocimiento.

Agradecemos a todos los profesores del departamento de Sistemas, al programa de ingeniería de sistemas, por darnos la posibilidad de estudiar esta hermosa carrera. También a la Facultad de ingeniería Electrónica (FIET) y sin duda a la Universidad del Cauca, por dejarnos hacer parte de esta comunidad unicaucana y brindarnos el apoyo para finalizar nuestra carrera.

Por último, queremos expresar nuestros agradecimientos a nuestras familias, por todo el amor, comprensión y colaboración en estos años, sin ustedes no hubiéramos logrado esta meta. Como agradecimiento especial queremos agradecer a nuestras madres, por estar ahí al frente, cuando se necesitaba y apoyarnos en todo.

¡¡¡Gracias!!!

Resumen estructurado

El procesamiento del lenguaje natural (PLN) es una de las áreas fuertemente investigadas en los últimos años, entre las tareas más importantes se encuentra el etiquetado de partes del discurso (Part-of-Speech Tagging, POST), la cual sirve en el preprocesamiento de la mayoría de aplicaciones de PLN. El etiquetado se ha abordado desde diferentes enfoques, pero es indispensable seguir buscando nuevos métodos más sencillos y eficientes, como el uso de algoritmos metaheurísticos, los cuales han mostrado ser superiores que otros métodos.

En esta tesis se propone abordar el problema de etiquetado desde una perspectiva de los algoritmos metaheurísticos, para ello en primera instancia, se propone la adaptación de tres algoritmos metaheurísticas al problema de etiquetado, evaluados sobre tres lenguas, dos lenguas tradicionales como lo son el Castellano e Inglés, y una no tradicional como el Nasa Yuwe, para el caso del Castellano se realizó el procesamiento de un corpus y se hizo la integración de los tres corpus en una sola base de datos. En segunda instancia, se propone una nueva versión memética con el algoritmo seleccionado anteriormente, buscando mejorar el desempeño, para ello se realizó la integración de una estrategia de búsqueda local al algoritmo, haciendo un balance entre exploración y explotación. Además, como una nueva estrategia para mejorar el etiquetado se presenta una mejora contextual para el caso del castellano, y, por último, se plasmó todos los resultados en el desarrollo de una aplicación web y un servicio web, donde se pueda realizar el etiquetado por parte de un usuario con los algoritmos presentados en este trabajo, el proceso se realizó bajo el desarrollo ágil Scrum.

El desarrollo de esta tesis estuvo enmarcado en la metodología Patrón de Investigación Iterativo, la cual permitió, en el primer ciclo, se enfocó en la adaptación de algoritmos metaheurísticos al problema de etiquetado y la selección del mejor; el segundo ciclo, en la propuesta de una nueva versión de un algoritmo memético; y, el

tercer ciclo, en el diseño y desarrollo de un prototipo software para el etiquetado de oraciones de los tres corpus con los algoritmos adaptados.

Como resultados concretos de esta tesis, se presenta en primer lugar, una breve reseña sobre el contexto y el estado de arte de los trabajos revisados. En segundo lugar, la definición de una línea base para el castellano que nos permitiera compararnos, con los nuevos algoritmos, la cual incluye la construcción de un dataset para realizar el etiquetado para la lengua castellana. En tercer lugar, se presenta la adaptación de dos algoritmos metaheurísticos al problema de etiquetado y los resultados sobre las lenguas Castellano, Inglés y Nasa Yuwe. En cuarto lugar, se presenta una nueva versión memética del algoritmo GBHS, además de otras mejoras y los resultados finales para cada lengua. Finalmente, se presenta el desarrollo de la aplicación web y servicio web, para el etiquetado de partes del discurso, y la publicación de los dos artefactos en la nube.

Los resultados obtenidos en el desarrollo de esta tesis, indican que el problema de etiquetado se puede seguir mejorando con nuevas técnicas de optimización y algoritmos más sencillos, por lo tanto, esta tesis se convierte en referente en seguir buscando nuevas estrategias y aplicaciones para el etiquetado de partes del discurso.

Palabras Clave: Identificador de partes del discurso, algoritmos metaheurísticos, algoritmo memético para etiquetado, conjunto de etiquetas, corpus etiquetado

CONTENIDO

Lista de figuras

Lista de tablas

Lista de ecuaciones

Lista de Algoritmos

Lista de anexos

Anexos Digitales

Anexo Digital 1_Scripts BDCorpus …….. shorturl.at/hjqrO

Anexo Digital 2_ Algoritmos POST …….. shorturl.at/mBDHU

Anexo Digital 3_ Configuración de experimentos …….. shorturl.at/cdgO3

Anexo Digital 4_Aplicación Web ……. shorturl.at/fgmuZ

Capítulo 1

Introducción

1.1 Presentación

El presente trabajo se desarrolló como tesis de pregrado de la Universidad del Cauca, para obtener el título de Ingeniero de Sistemas, abordando como temas principales: 1) El etiquetado de partes del discurso POST desde una perspectiva metaheurística. 2) El uso de tres corpus etiquetados, dos para lenguas tradicionales (uno para Inglés y otro para Castellano) y uno para una lengua no tradicional (Nasa Yuwe) y 3) Un Conjunto de etiquetas para los corpus mencionados anteriormente.

1.2 Planteamiento del problema

El procesamiento del lenguaje natural (PLN) es un área de investigación que estudia como las aplicaciones informáticas pueden ser usadas para entender, interpretar y manipular el lenguaje humano (por ejemplo, un texto) [1]. Una de las tareas más importantes de preprocesamiento para la mayoría de las aplicaciones de PLN, es el etiquetado de partes del discurso (Part-of-speech Tagging, POST) [2]. Algunas áreas de aplicación del POST incluyen: El análisis de sentimientos [3], donde el etiquetado reconoce diferentes partes de un texto y ayuda en la determinación de polaridad[1] [4]; Las máquinas de traducción[2], en donde el etiquetado es crucial para una traducción de alta calidad, dado que proporciona características importantes de la fragmentación y el análisis de los textos [5]; El reconocimiento de voz [6], donde el etiquetado es un paso preliminar para explotar las transcripciones de audio[3], aportando conocimiento lingüístico, por ejemplo, información morfosintáctica, que genera mejores transcripciones; y en tareas de recuperación de información [7], donde el etiquetado ayuda en el post-procesamiento, dado que las palabras etiquetadas como sustantivos, pueden clarificar el rol de cada término en consultas y documentos.

El proceso de etiquetado de partes del discurso, consiste en la asignación de una etiqueta a cada palabra dentro de un texto, la cual representa la categoría léxica de la palabra [8]. Por ejemplo, la palabra “el” se puede etiquetar como “articulo” o

[1] Determina si el sentimiento del texto es positivo, negativo o neutral.

[2] Las máquinas de traducción automática o automatizada, trasladan (o “traducen”) la semántica de un texto escrito en un idioma desconocido para el usuario, a la semántica de un idioma conocido por el usuario [91]

[3] Es la conversión de una secuencia de palabras en un texto a señales de audio [6]

“preposición” según el lugar en el que se encuentre dentro de la oración y las palabras que la preceden o la suceden, de tal forma que el etiquetado se convierte en una tarea compleja, al enfrentar principalmente tres grandes retos [8,9] a saber: 1) La ambigüedad de palabras, que puede ocasionar un etiquetado incorrecto, dadas las múltiples etiquetas que puede tener cada palabra, por ejemplo, en la oración “*Mike has the can*” la palabra “*can*” se etiqueta como sustantivo, mientras que en “*They can run fast*” la misma palabra se etiqueta como verbo. 2) El tamaño del conjunto de etiquetas, un conjunto de etiquetas muy especializado puede llegar a dificultar la distinción entre varias etiquetas similares y un conjunto de etiquetas muy general, puede ocasionar pérdida de conocimiento morfológico y morfosintáctico para las palabras a etiquetar ocasionando un etiquetado incorrecto. 3) El etiquetado de palabras desconocidas, es decir, palabras que no se encuentran en los datos de entrenamiento, o sobre las cuales no se tienen previamente definidas reglas para su etiquetado, por tanto, se dificulta asignar automáticamente la etiqueta correcta.

Para la construcción de los etiquetadores existen varios enfoques, entre ellos se encuentran: 1) Los basados en reglas [10], que consisten en asignar una regla a una palabra, a partir de esto, se aplica una transformación de reglas hasta que la palabra se ajuste a la etiqueta más adecuada [11], estos etiquetadores tienen como principal inconveniente, que se requiere conocimiento específico de la lengua a etiquetar, para la generación de las reglas. 2) Los basados en información estadística [12], que buscan encontrar la secuencia de etiquetas más probable para cada oración, es decir, para etiquetar una palabra en la oración, tienen en cuenta la(s) etiqueta(s) de su contexto[4] y la probabilidad de que estas ocurran. Estas probabilidades pueden calcularse directamente sobre un corpus previamente etiquetado (datos de entrenamiento); estos etiquetadores no requieren conocimiento de reglas específicas del lenguaje [13], entre las técnicas más reconocidas se incluyen los modelos ocultos de Markov [14], los modelos de máxima entropía [15] los trigramas [16], las máquinas de soporte vectorial [17] y las redes neuronales [18].

Estos enfoques tradicionales han mostrado buenos resultados, no obstante, existen nuevas propuestas que abordan el problema de etiquetado utilizando algoritmos metaheurísticos, tanto desde una perspectiva estadística como basada en reglas. Estas propuestas metaheurísticas han obtenido un desempeño sobresaliente en comparación con los enfoques descritos anteriormente, como es el caso de Araujo *et al*. [19] que utilizan un algoritmo genético, el recocido simulado y el algoritmo CHC, para el problema de etiquetado buscando determinar cuál es el mejor, mediante una

[4] Es decir, la etiqueta de la(s) palabra(s) predecesora(a) o sucesora(s) [24]

evaluación sobre el corpus Brown [20] para el Inglés. Silva *et al*.[21], que abordan el etiquetado como un problema de optimización usando el algoritmo de optimización por enjambre de partículas (Particle Swarm Optimization, PSO), trabajando sobre dos corpus etiquetados para el Inglés (Corpus Brown [20] y Penn Treebank [22]) y además un corpus para el portugués (corpus Mac-Morpho [23]). Forsati *et al*. [24] que utilizan la metaheurística de la Búsqueda Armónica (Harmony Search, HS), abordando el problema de etiquetado desde un enfoque estadístico aplicado a los corpus Brown y Penn Treebank. Sierra *et al*. [25] que proponen un algoritmo memético para el problema de POST, mediante la adaptación de la metaheurística Global-Best Harmony Search (GBHS)[5] y la inclusión de conocimiento del problema, obteniendo muy buenos resultados sobre el corpus Brown. Sierra *et al.* [26] que evaluaron el algoritmo memético presentado en [25] y otras metaheurísticas sobre un corpus construido para la lengua Nasa Yuwe (lengua Páez de una etnia de indígenas del sur occidente de Colombia), obteniendo como resultado que la propuesta memética basada en GBHS de [25] obtiene mejores valores de precisión (medida comúnmente usada para evaluar la calidad del POST) en la asignación de las etiquetas.

No obstante, cabe destacar que:

- En [26] se presenta un buen aporte para la lengua Nasa yuwe, pero no es un valor de precisión adecuado para implementar una herramienta software que se dedique a esta tarea, por tanto, es crucial buscar mejores valores de precisión para esta lengua.
- No se cuenta con un algoritmo metaheurístico para etiquetado que provea buenos resultados en las tres lenguas seleccionadas (castellano, Nasa Yuwe e inglés) para este trabajo.
- Recientemente en la literatura, las técnicas de optimización evolutiva se continúan considerando como herramientas efectivas para problemas de optimización de variables discretas y continuas, como es el caso de Jaya [27], que se presenta como un algoritmo más fácil de usar que otros algoritmos evolutivos y de estado simple; este algoritmo encuentra soluciones óptimas más rápidas debido al balance que logra entre explotación y exploración, además de no requerir parámetros, lo cual es crucial para la simplicidad de una solución [28].
- Cada una de las propuestas mencionadas anteriormente y en el estado del arte han demostrado buenos resultados, todas caracterizándose por ser soluciones más simples que los enfoques basados en reglas e información estadística, no obstante, existen oportunidades de encontrar mejores soluciones con algoritmos

[5] Global Best Harmony Search (GBHS) es una metaheurística basada en la búsqueda armónica (HS) e inteligencia de enjambre [32] que supera en rendimiento a la búsqueda armónica [53].

metaheurísticos haciendo uso de las diferentes características de cada uno o versiones meméticas que además de la exploración y explotación del espacio de búsqueda incorporen conocimiento específico del problema, para que el problema de etiquetado sea cada día de mayor calidad.

Por lo anteriormente presentado, es primordial tener en cuenta:

- El segundo teorema de los No Free Lunch Theorem (NFLT) [29] (Un algoritmo metaheurístico puede llegar a superar a otro en un determinado tipo de problema, aun cuando ninguno de ellos este especializado en dicho problema[6]), soportó la necesidad de seguir buscando otras opciones para resolver el problema de etiquetado, que se caractericen por ser soluciones sencillas, tener buena calidad medida en precisión y especializarse en este problema, para las lenguas de interés de esta propuesta.
- Los buenos resultados obtenidos por el único memético adaptado al problema de etiquetado [25] [26], motivaron a buscar nuevas estrategias de inclusión de conocimiento para este algoritmo o para algoritmos como es Jaya [27] y PSO [30].

Por lo tanto, en este proyecto, se planteó la siguiente pregunta de investigación: **¿Es posible obtener mejores resultados (medidos en precisión) para el problema de etiquetado de partes del discurso, mediante propuestas que involucren el uso de algoritmos metaheurísticos o meméticos?**

Para dar solución a la anterior pregunta, se propuso dar continuidad a lo propuesto en [25] y [26], adaptar los algoritmos metaheurísticos Jaya [27] y PSO [30] al problema de etiquetado, con el fin de comparar y seleccionar el mejor algoritmo, para posteriormente incluir optimizadores locales con conocimiento específico del problema, convirtiéndolo así en propuestas meméticas; todas las versiones propuestas fueron evaluadas sobre los corpus etiquetados de lenguas tradicionales como el castellano y el inglés, de igual forma sobre el corpus de una lengua no tradicional e independiente como el Nasa Yuwe [31], en pro de evaluar el comportamiento de los algoritmos en diferentes contextos lingüísticos.

En esta investigación se decidió utilizar la metaheurística simple Hill Climbing como optimizador local en el algoritmo memético adaptado al problema de etiquetado, debido a que esta se mueve de acuerdo a su vecindario (neighbords), favoreciendo el etiquetado, ya que el cambio de una etiqueta predecesora o sucesora puede mejorar la solución en pocas iteraciones, reduciendo a su vez el número de evaluaciones de la función objetivo, además, ésta metaheurística se utilizó en un algoritmo memetico para

[6] Interpretación del teorema [92]

el problema de etiquetado [25], mostrando un buen desempeño. También se decidió revisar el algoritmo Random Restart Hill Climbing debido a que es una mejora del HC, que soluciona el problema de quedarse atrapado en óptimos locales, lo cual es prometedor en la aplicación al problema de etiquetado.

1.3 Objetivos

A continuación, se presentan los objetivos como fueron aprobados por el Consejo de Facultad de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca en el documento de la propuesta de tesis, que buscan resolver lo presentado en la sección anterior, planteamiento del problema.

1.3.1 Objetivo general

Construir un prototipo software para etiquetar las partes del discurso de oraciones en Inglés, Castellano y Nasa Yuwe, basado en la adaptación de un algoritmo metaheurístico y su versión memética para el problema de etiquetado, buscando mejorar los resultados de precisión presentados en las referencias [25] y [26], al mismo tiempo que se pueda aprovechar la sencillez de las soluciones propuestas por los algoritmos metaheurísticos.

1.3.2 Objetivos específicos

- Adaptar al problema de etiquetado los algoritmos metaheurísticos[7] Jaya [27], PSO [30] y GBHS [25], con el fin de seleccionar el mejor algoritmo para este problema, mediante:
  - El ajuste de las características de las soluciones, la función objetivo y los parámetros de cada algoritmo al problema de etiquetado, y
  - La realización de un experimento que permita evaluar el desempeño de los tres algoritmos adaptados sobre conjuntos de datos de los corpus Brown [19] en Inglés, IULA [56] en castellano y el Corpus etiquetado de Nasa Yuwe [25], utilizando validación cruzada y la medida de precisión para su comparación.

- Modelar una versión memética del mejor algoritmo metaheurístico seleccionado en el primer objetivo, mediante la integración de técnicas de optimización local (Hill Climbing [32] y Hill Climbing con reinicios aleatorios [33]) y conocimiento específico

[7] Se delimita solo a tres algoritmos debido a que en el proceso de adaptación de las metaheurísticas al problema de etiquetado se pueden presentar muchas posibilidades y esta propuesta solo cuenta con 9 meses.

del problema, aplicadas al etiquetado de partes del discurso de los corpus Brown [19], IULA [56] y Nasa Yuwe [25] y evaluándolo de la misma forma que en el objetivo anterior.

- Diseñar e implementar un servicio web y una aplicación web adaptativa, para etiquetar oraciones de inglés, castellano y Nasa Yuwe utilizando el algoritmo metaheurístico seleccionado, su versión memética y los algoritmos comparados, haciendo uso del proceso de desarrollo ágil SCRUM [34].

1.4 Contribuciones de esta tesis

A nivel de investigación formativa e innovación, se obtuvo una línea base para posteriores trabajos en algoritmos metaheurísticos y meméticos para la identificación de partes del discurso y sus aplicaciones, mediante: 1) La adaptación e implementación de cada una de las metaheurísticas Jaya [27], PSO [30] y GBHS [25] al problema de etiquetado y sus correspondientes evaluaciones sobre 2 corpus etiquetados de lenguas tradicionales y 1 corpus de una lengua no tradicional. 2) La mejora de la versión memética propuesta por [25] aplicando el uso de diferentes contextos de la palabra a etiquetar 3) El modelado de una versión memética de la mejor metaheurística adaptada al problema de etiquetado y su evaluación sobre los mismos corpus del ítem anterior. 4) El establecimiento de otro antecedente en investigación en el área de computación lingüística sobre la lengua Nasa Yuwe, el cual permita impulsar nuevas investigaciones o aplicaciones.

A nivel de desarrollo tecnológico, se diseñó e implementó: 1) Un servicio web con las adaptaciones de los algoritmos metaheurísticos Jaya [27], PSO [30] y GBHS [25] al problema de etiquetado y la correspondiente versión memética modelada del mejor seleccionado. 2) Una aplicación web para etiquetar oraciones en las lenguas Castellano, Inglés y Nasa Yuwe, que consume el servicio web creado.

Otro aporte es el procesamiento del corpus IULA de la lengua castellana con sus respectivas frases y términos, que fue almacenado en una base de datos para llevar a cabo el etiquetado, además se hizo la integración en una sola base de datos de los corpus IULA, Brown y Nasa Yuwe, que permitiera a los algoritmos implementados ejecutar el etiquetado para cada lengua.

A nivel social, se espera que los resultados de esta propuesta puedan ser utilizados en diferentes aplicaciones o actividades, en pro de apoyar la revitalización de la lengua Nasa y ser una base para continuar con la visibilización de ésta.

1.5 Diseño metodológico utilizado

Como metodología para el desarrollo de este proyecto, en los dos primeros objetivos específicos, se utilizará el Patrón Iterativo de Investigación (Iterative Research Pattern, IRP) propuesto por Pratt [35], que es un método diseñado especialmente para proyectos de investigación de ciencias de la computación y que involucran una solución computacional. Este proceso se enfoca en múltiples y cortos ciclos con 4 pasos básicos: observación del problema, identificación del problema, desarrollo tecnológico de la solución y pruebas de la solución. Adicionalmente, se utilizará el proceso de desarrollo de software SCRUM [34] para apoyar el desarrollo e implementación del objetivo específico 3. Por tanto, el plan de actividades se estructuró en 2 paquetes de trabajo (Work Package, WP) [36] así:

1. **Paquete de trabajo 1**: Corresponde a las actividades enmarcadas en el desarrollo de los objetivos específicos 1 y 2:

1.1. Observación

Ciclo 1: En este ciclo las actividades fueron: *Actividad 1.1.1***:** Profundización en metaheurísticas, meméticos e identificación de partes del discurso. *Actividad 1.1.2*: Revisión del estado del arte sobre etiquetadores, que aborden el problema desde una perspectiva metaheurística. *Actividad 1.1.3*: Implementación de los algoritmos metaheurísticos Jaya, PSO y GBHS.

Ciclo 2: En este ciclo las actividades realizadas: *Actividad 1.1.4*: Profundización en metaheurísticas de búsqueda local (Hill Climbing y Hill Climbing con reinicios aleatorios).

1.2. Identificación

Ciclo 1: En este paso se encuentran las siguientes actividades: *Actividad 1.2.1*: Identificación de características para la adaptación de los algoritmos metaheurísticos en la construcción de los etiquetadores.

Ciclo 2: *Actividad 1.1.5*: Diseño de estrategias de búsqueda local (Hill Climbing y Hill Climbing con reinicios aleatorios) e inclusión de conocimiento específico del problema.

1.3. Desarrollo de la aplicación

Ciclo 1: En este ciclo se encuentran las siguientes actividades: *Actividad 1.3.1:* Adaptación e implementación de los algoritmos metaheurísticos al problema de etiquetado.

Ciclo 2: En este ciclo se encuentran las siguientes actividades: *Actividad 1.3.2*: Adaptación e implementación de la(s) estrategia(s) de búsqueda local Hill Climbing al mejor algoritmo metaheurístico seleccionado. *Actividad 1.3.3*: Adaptación de la estrategia de búsqueda local Hill Climbing con reinicios aleatorios al mejor algoritmo

metaheurístico seleccionado. En cada una de las actividades se incluye el manejo de conocimiento específico del problema para realizar la explotación del vecindario.

1.4. Pruebas

Ciclo 1: *Actividad 1.4.1:* Procesamiento de los corpus lingüísticos (Inglés, Castellano y Nasa Yuwe). *Actividad 1.4.2:* Configuración del experimento: definición y ajuste de los parámetros a utilizar en los algoritmos metaheurísticos implementados. *Actividad 1.4.3:* Evaluación de los algoritmos metaheurísticos sobre el corpus en Inglés. *Actividad 1.4.4:* Evaluación de los algoritmos metaheurísticos sobre el corpus en Castellano. *Actividad 1.4.5:* Evaluación de los algoritmos metaheurísticos sobre el corpus en Nasa Yuwe. *Actividad 1.*4.6: Análisis de resultados y selección del mejor algoritmo metaheurístico basado en el desempeño de los algoritmos sobre los corpus mencionados, teniendo en cuenta los valores de la medida de precisión

Ciclo 2: En este ciclo se encuentran las siguientes actividades: *Actividad 1.4.7:* Configuración del experimento: definición y ajuste de los parámetros a utilizar en el algoritmo memético implementado. *Actividad 1.4.8:* Realizar la evaluación del algoritmo memético sobre sobre el corpus en Inglés. *Actividad 1.4.9:* Realizar evaluación del algoritmo memético sobre el corpus en Castellano. *Actividad 1.4.10:* Realizar evaluación del algoritmo memético sobre el corpus en Nasa Yuwe.

2. **Paquete de trabajo 2**: Corresponde a las actividades enmarcadas en el desarrollo del objetivo específico 3.

Actividad 2.1: Obtención de la lista de producto y definición de la lista de pendientes para el sprint 1, se utilizará como artefacto las historias de usuarios para la captura de requisitos.

Sprint 1: Diseño e implementación del servicio web con las siguientes actividades: *Actividad 2.2.1:* Realizar SCRUM diario para evaluar el progreso del sprint 1. *Actividad 2.2.2:* Realizar los artefactos de análisis y diseño del servicio web que permitan definir la arquitectura, las clases, las colaboraciones y responsabilidades, que satisfagan los requisitos establecidos en las historias de usuario. *Actividad 2.2.3:* Realizar la codificación de cada historia de usuario de la lista de pendientes del sprint 1. *Actividad 2.2.4:* Realizar diseño y ejecución de pruebas de unidad e integración de los requisitos implementados en el sprint 1. *Actividad 2.2.5:* Realizar la revisión del sprint 1 para inspeccionar y establecer el incremento. *Actividad 2.*2.6: Establecer el estado del incremento del sprint 1. *Actividad 2.*2.7: Realizar la retrospectiva del sprint 1. *Actividad 2.*2.8: Despliegue del servicio web implementado.

Actividad 2.3: Obtención de la lista de producto y definición de la lista de pendientes para el sprint 2.

Sprint 2: Diseño e implementación de la aplicación web con las siguientes actividades: *Actividad 2.4.1:* Realizar SCRUM diario para evaluar el progreso del sprint 2. *Actividad 2.4.2:* Realizar los artefactos de análisis y diseño de la aplicación web que permitan definir la arquitectura, las clases, las colaboraciones y responsabilidades, que satisfagan los requisitos establecidos en las historias de usuario. *Actividad 2.4.3:* Realizar la codificación de cada historia de usuario de la lista de pendientes del sprint 2. *Actividad 2.4.4:* Realizar diseño y ejecución de pruebas de unidad e integración de los requisitos implementados en el sprint 2. *Actividad 2.4.5:* Realizar la revisión del sprint 2 para inspeccionar y establecer el incremento. *Actividad 2.4.6:* Establecer el estado del incremento del sprint 2. *Actividad 2.4.7*: Realizar la retrospectiva del sprint 2. *Actividad 2.4.8*: Realizar pruebas de usuario sobre la aplicación web adaptativa implementada.

Actividad 2.5: Análisis y documentación del articulo y la monografía.

1.6 Organización del documento

En esta sección se presentó: 1) el planteamiento del problema; 2) los objetivos y 3) las contribuciones a nivel de investigación que se obtuvieron con el desarrollo de este trabajo de grado.

En el capítulo 2, se presenta el estado del arte, que incluye un contexto teórico y la revisión de los trabajos más relevantes relacionados con los temas tratados en este trabajo.

Después, en el capítulo 3, se hace el procesamiento de un corpus de la lengua castellana, además de la construcción de un dataset, para llevar a cabo los experimentos sobre esta lengua; también se hace la adaptación de dos algoritmos metaheurísticos al problema de etiquetado.

En el capítulo 4, se realiza una nueva versión memética para el problema de etiquetado, además de otras adaptaciones de algoritmos metaheurísticos al problema de etiquetado, que mejoran la precisión.

En el capítulo 5, se desarrolla un servicio web y una aplicación web, que permiten realizar el etiquetado por parte de un usuario.

Finalmente, en el capítulo 6, se presentan las conclusiones alcanzadas con la realización de este trabajo y las propuestas de trabajos futuros.

Capítulo 2

Estado del Arte

2.1. Contexto

En este capítulo, se presentan algunos de los conceptos más relevantes en el desarrollo de esta investigación, como son: identificación de partes del discurso, corpus etiquetados, conjuntos de etiquetas, algoritmos metaheurísticos y algoritmos meméticos. Además, se presenta la definición de los algoritmos trabajados en esta investigación.

2.1.1 Identificación de partes del discurso

Los etiquetadores o identificadores de partes del discurso (Part-of-Speech Tagging, POST) utilizan el contexto (relación entre palabras adyacentes) de una palabra para asignar una correspondiente etiqueta, son una herramienta utilizada por muchas aplicaciones del procesamiento del lenguaje natural, debido a la información que provee, entre ellas están: reconocimiento de frases, desambiguación del sentido de palabras y asignación de funciones gramaticales [37].

En la identificación de partes del discurso se presentan algunas dificultades [8,9] cuando se enfrenta a: a) Un conjunto de etiquetas muy grande, que a pesar de exponer más información sobre la estructura de una palabra, es más difícil asignarle adecuadamente una etiqueta, b) Palabras ambiguas, se presenta cuando una oración contiene palabras para las cuales existe más de una posible etiqueta, el etiquetador debe ser capaz de desambiguar la palabra, con información de otras palabras en la oración c) Palabras desconocidas, según el enfoque son palabras que no se encuentran en los datos de entrenamiento o no existe una regla para ellas.

Existen dos principales enfoques para la construcción de etiquetadores, los basados en información estadística y los basados en reglas. En la **Figura 1** se muestran los enfoques, con una breve descripción y algunos de sus métodos.

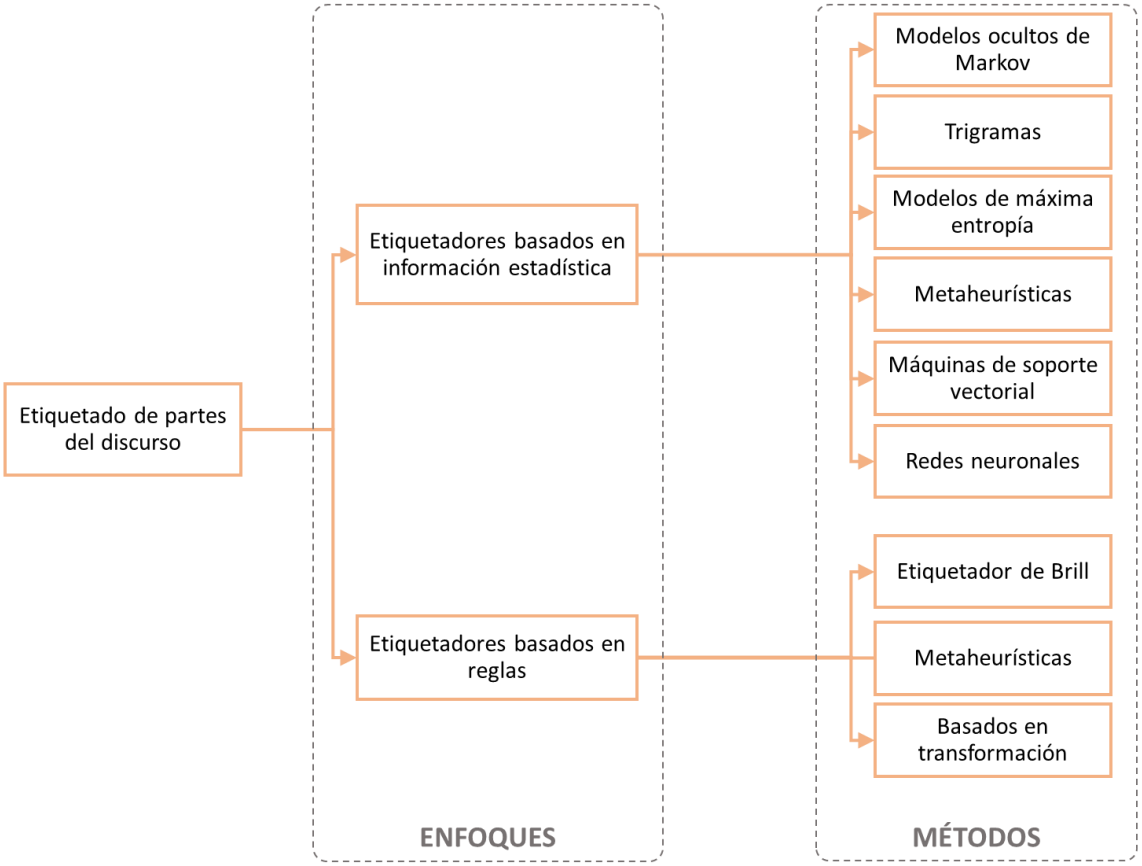


Figura 1. Principales enfoques de etiquetado

Los enfoques estadísticos resuelven el problema de etiquetado mediante el uso de un corpus etiquetado que ayuda en la identificación de la secuencia de etiquetas más probables, basado en la ocurrencia estadística del contexto de la frase y la frecuencia de la palabra a etiquetar [24]. La investigación sobre estos métodos ha crecido mucho en los últimos años debido a su simplicidad, independencia del idioma y disponibilidad de grandes corpus etiquetados [38]. Entre los principales métodos estadísticos se encuentran los modelos ocultos de Markov [14], los modelos de máxima entropía [15], los trigramas [16], las máquinas de soporte vectorial [17] y las redes neuronales [18].

Los enfoques basados en reglas extraen y almacenan información de la estructura de un lenguaje en forma de reglas linguísticas, su principal función es asignar una regla a una palabra (una regla esta conformada por información sintáctica y morfológica de cada palabra), a partir de esto, se aplica una transformación de reglas hasta encontrar la etiqueta (regla) más adecuada; en el proceso de etiquetado, una palabra puede tener una lista de reglas, para lo cual, el etiquetador requiere de un analizador léxico que resuelve las ambigüedades [10], el principal inconveniente que tienen estos enfoques es que se requiere conocimiento específico de la lengua a etiquetar, para la

generación de las reglas. Entre los trabajos más reconocidos se encuentra el etiquetador de Brill [10] para el inglés, que se constituye como base de otros enfoques y propuestas.

Actualmente se destaca el uso de algoritmos metaheurísticos para la construcción de etiquetadores aplicado a enfoques basados en reglas y basados en información estadística; estos algoritmos modelan el problema de etiquetado como un proceso de búsqueda complejo, haciéndolo más eficiente y menos costoso que los enfoques tradicionales [24], por esta razón, este proyecto se delimitó al uso de algoritmos metaheurísticos aplicado a enfoques estadísticos (debido a su simplicidad). En la sección de trabajos relacionados se describen los trabajos más relevantes de algoritmos metaheurísticos aplicados al problema de etiquetado.

2.1.2 Algoritmos metaheurísticos y meméticos

Las metaheurísticas son conocidas como enfoques eficientes para problemas de optimización complejos, son aplicadas a problemas donde no se requiere de antemano encontrar una solución óptima y la búsqueda por fuerza bruta está descartada debido a que el espacio de soluciones es demasiado grande [39]. Las metaheurísticas han ganado más popularidad que los métodos exactos en resolver problemas de optimización, debido a su simplicidad y a la solidez de sus resultados; además han sido implementadas en diferentes campos como la ingeniería, negocios, transporte e incluso ciencias sociales [28].

El proceso de búsqueda de un algoritmo metaheurístico es llevado a cabo por múltiples agentes que forman un sistema de soluciones, estos evolucionan utilizando un conjunto de reglas o ecuaciones matemáticas durante múltiples iteraciones; las iteraciones continúan hasta que la solución encontrada cumple algún criterio predefinido; la solución final es considerada la más óptima del sistema [28].

Los algoritmos metaheurísticos combinan la capacidad de búsqueda local y búsqueda global, a fin de mantener un balance entre exploración y explotación, una metaheurística tendrá éxito en un problema de optimización en la medida en que este equilibrio se logre. La exploración se encarga de diversificar las soluciones, mientras que la explotación intensifica la búsqueda, identificando partes del espacio de búsqueda con soluciones de alta calidad [39].

Metaheurísticas de estado simple: también son llamados métodos de trayectoria, éstas comienzan con una única solución inicial y se alejan de ella describiendo una trayectoria en el espacio de búsqueda [39], algunas de estas metaheurísticas son:

recocido simulado, búsqueda tabú, método GRASP, hill climbing y hill climbing con reinicios aleatorios.

Metaheurísticas poblacionales: en contraste con las de estado simple, tratan con un conjunto de soluciones (población) en lugar de una única solución; las metaheurísticas más estudiadas se encuentran relacionadas con computación evolutiva (Evolutionary Computation - EC) e inteligencia de enjambre (Swarm Intelligence - SI); los algoritmos de EC, modifican la población a través de operadores de recombinación y mutación, entre ellos se encuentran: algoritmo genético, programación evolutiva, programación genética y evolución diferencial, mientras que los SI tratan de proporcionar inteligencia computacional, entre ellos se encuentran: optimización por enjambre de partículas (PSO), optimización de enjambre de hormigas (ACO) y algoritmos basados en la optimización de colonia de abejas (BCO).

Otra familia de metaheurísticas son los algoritmos meméticos, los cuales intentan aunar ideas y conceptos de diferentes técnicas de resolución, por ejemplo, los algoritmos evolutivos y la búsqueda tabú [40]. Un algoritmo memético es una estrategia de búsqueda cuyas soluciones candidatas (o agentes) compiten (búsqueda local) y cooperan (búsqueda global) de manera sinérgica, además estos agentes hacen uso de explícito de conocimiento sobre el problema que se quiere resolver [25].

A continuación, se detallan algunos algoritmos metaheurísticos de estados simples y poblacionales, utilizados en este trabajo.

El algoritmo Hill Climbing HC, es una estrategia de búsqueda local, este genera una solución candidata en cada iteración, moviéndose continuamente en el espacio de soluciones, con el fin de encontrar la mejor solución al problema, este algoritmo finaliza cuando encuentra el óptimo local. El principal problema al que se enfrenta son los óptimos locales, debido a que no tiene la capacidad de explorar otros espacios de búsqueda [32]; el pseudocódigo del HC se muestra en el **Algoritmo 1**.

Algoritmo 1. Pseudocódigo Hill Climbing

1. Inicializar la solución inicial $s, iterations\_HC$
2. fitness (s)
3. $i\_restart = 0$
4. $s' = s$
5. **for** $i = 0$ to $iterations\_HC$
6. $s' \leftarrow$ one new neighbors generated from s
7. **if** (fitness (s') > fitness (s))
8. $s \leftarrow s'$
9. **end for**

Fuente: [41]

El algoritmo Random Restart Hill Climbing RRHC [42], es una mejora basada en HC que busca resolver el problema de los óptimos locales, esto implica realizar exploraciones repetitivas en el espacio del problema, generando estados iniciales de forma aleatoria hasta que se detiene o no se encuentra una mejor solución. Esto permite la comparación de muchos casos pruebas, de esta forma, encontrar una solución óptima es cuestión de usar suficientes iteraciones (reinicios). En el **Algoritmo 2** se muestra el pseudocódigo de la metaheurística [32][43].

Algoritmo 2. Pseudocódigo Restart Random Hill Climbing

1. Inicializar la solución inicial $s, iterations\_HC$ y $n\_restart$
2. fitness (s)
3. $i\_restart = 0$
4. $s' = s$
5. **for** $i = 0$ to $iterations\_HC$
6. **if** $i\_restart > n\_restart$ reinicie la solución inicial s a partir de s'
7. $s \leftarrow$ new solution
8. $i\_restart = 0$
9. **End**
10. $s' \leftarrow$ one new neighbors generated from s
11. **if** (fitness (s') fitness (s))
12. $s \leftarrow s'$
13. **else**
14. $i\_restart++$
15. **end**
16. **end for**

Fuente: [44]

El algoritmo Jaya [45], es presentado como un algoritmo simple, que únicamente requiere parámetros de control comunes; es considerado muy fácil de usar en contraste con otros algoritmos evolutivos, debido a que este algoritmo encuentra soluciones más óptimas, además de no requerir parámetros de control específicos [28]. En su naturaleza tiene como objetivo principal acercarse al éxito, es decir, encontrar la más óptima de las soluciones en el menor tiempo, pero también a su vez siempre está intentando alejarse del fracaso, es decir, de la peor de las soluciones, generando un balance óptimo entre exploración y explotación. En el estado del arte no se evidencia trabajos relacionados con la aplicación de Jaya al problema de etiquetado. El pseudocódigo de Jaya se presenta en el **Algoritmo 3**.

Algoritmo 3. Pseudocódigo JAYA

1. Inicializar tamaño de la población P
2. Inicializar número de variables de diseño
3. Inicializar criterio de finalización
4. Inicializar el número de variables de diseño
5. Definir el criterio de finalización $CriterioFinalizacion$
6. $s \leftarrow P$
7. **for** $i = 0$ to i_max

Algoritmo 3. Pseudocódigo JAYA

```
        gBest = best p in P
        gWorse = worse p in P
        s' ← definido por la Ecuación 1

        if fitness (s') > fitness (s)
            s ← s'

        if (CriterioFinalizacion)
            return s
    end
```

Fuente: [45]

$$X'_{j,k,i} = X_{j,k,i} + r_{1,j,i}\left[\left(X_{j,best,i}\right) - \left|\left(X_{j,k,i}\right)\right|\right] - r_{2,j,i}\left[\left(X_{j,worst,i}\right) - \left|\left(X_{j,k,i}\right)\right|\right]$$

Ecuación 1. Algoritmo Jaya

El algoritmo Particle Swarm Optimization PSO fue propuesto por Kennedy y Eberhart [46], es una de las técnicas de optimización más utilizadas que ha obtenido muy buenos resultados de una manera más rápida y económica que otros métodos [43]. PSO ha sido aplicado con éxito a una amplia gama de aplicaciones, como la optimización de funciones, la asignación de tareas y los problemas de programación [47]; el algoritmo original de [46] está diseñado para resolver problemas de optimización continuo, esta ha sido una de las cuestiones para el diseño del algoritmo a problemas discretos, debido a que las ecuaciones del PSO original no se pueden usar para generar una solución discreta, ya que las posiciones que se generan tienen un valor real [48]; sin embargo, existen propuestas que han diseñado el algoritmo PSO discreto dependiendo del problema a optimizar, algunos son: Pan *et al* en [48], Boussaid *et al* en [49] y Crec *et al* en [50]. El algoritmo PSO, es una metaheurística poblacional motivada por el comportamiento colectivo inteligente de algunos animales. Cada solución potencial se denomina partícula, el conjunto de partículas es conocido como enjambre, donde la posición de cada partícula cambia en función de su propia experiencia (pBest) y la experiencia del enjambre (gBest) [51]; en cada iteración la velocidad y la posición son actualizados siguiendo las ecuaciones descritas en el **Algoritmo 4**.

Algoritmo 4. Pseudocódigo Particle Swarm Optimization (PSO)

```
1.  Inicializar enjambre P
2.  for i = 0 to i_max
3.      foreach p in P
4.          if fitness(p) > fitness(pBest)
5.              pBest ← p
6.          end
7.      end
8.      gBest = best p in P
```

Algoritmo 4. Pseudocódigo Particle Swarm Optimization (PSO)

```
9.      foreach p in P
10.         v ← v + c1 * rand * (pBest − p) + c2 * rand * (gBest − p)
11.         p ← p + v
12.     end
13. end
```

Fuente: [52]

El algoritmo memético GBHS Tagger presentado en [25], está basado en las metaheurísticas global best harmony search (GBHS) como estrategia de optimización global y Hill Climbing (HC) como el optimizador local; en cada iteración del algoritmo se aplica GBHS y el proceso de optimización local a la mejor improvisación de la memoria armónica, de acuerdo a la probabilidad $ProbOpt$. El algoritmo GBHS es inspirado por el concepto de inteligencia de enjambre propuesto en el algoritmo PSO y es una variación del algoritmo Harmony Search (HS), el GBHS modifica el paso de ajuste de tono de HS de modo que la nueva armonía pueda imitar la mejor armonía en la memoria armónica (HMS), esta modificación le permite al GBHS trabajar eficientemente en problemas continuos y discretos [53]; el pseudocódigo del memético GBHS Tagger, se presenta en el **Algoritmo 5**.

Algoritmo 5. Memético Global Best harmony Search (GBHS) y Hill Climbing

```
1.  Definir parámetros: HMS, NI, HCMR, PARMin, ParMax, ProbOpt, Alpha, MaxNeighbors
2.  Inicialización aleatoria de HMS o inicialización mejorada usando el parámetro Alpha
3.  foreach i = 1 to N
4.      PAR ← PARMin + (PARMax − PARMin) × (i / NI)
5.      if (U(0,1) ≤ HMCR)
6.          X'_i = X_i^j donde j~U(1, ..., HMS)
7.              if (U(0,1) ≤ PAR)
8.                  X'_i = X'_i ± r × bw, donde r~U(0,1)
9.              end
10.     else /*selección aleatoria*/
11.         X'_i = LB_i + r × (UB_i − LB_i)
12.     end
13.     if (fitness (X'_i)>fitness de la peor armonía en HMS)
14.         HMS[pos_peor] ← X'_i
15.     end
16.     if (U(0,1) < ProbOpt)
17.         Aplicar Optimización Local a la mejor armonía en HM
18.     end
19. end
```

2.2 Estado del arte

Dado que la temática de esta propuesta está delimitada al uso de algoritmos metaheurísticos y meméticos en el problema de POST, se realizó una revisión en las siguientes bases de datos y motores de búsqueda: IEEE, Springer, ACM, Scopus, ScienceDirect, Google Académico y ProQuest, que permitió encontrar: 1) El problema de etiquetado está siendo cada vez más explorado desde diferentes enfoques. 2) El uso de algoritmos metaheurísticos en este problema, ha ido tomando fuerza en los últimos años (2010-2020). 3) El idioma inglés y el conjunto de etiquetas Universal son los más utilizados para la evaluación de desempeño de los etiquetadores.

Las cadenas de búsqueda utilizadas en la revisión fueron las siguientes: a) *Metaheuristic algorithm + part of Speech Tagging POST, Part-Of-Speech Tagging or Tagger, Metaheuristics for natural language tagging, memetic algorithm + Part of Speech Tagging or POST,* con 16 artículos encontrados. b) *Tagged Corpus labeled corpus y POST, annotated corpus y POST, Part-Of-Speech Tagging,* con 11 artículos. c) *Tagset POST, conjunto de etiquetas*, con 19 artículos.

A continuación, se describen brevemente algunos de los trabajos encontrados:

2.2.1. Etiquetadores basados en metaheurísticas

En 2018, Alhasan y Al-taani [54] presentaron un etiquetador de partes del discurso para el idioma árabe utilizando el algoritmo de optimización por colonia de abejas (Bee Colony Optimization, BCO). El problema es representado como un grafo, en donde las posibles etiquetas de una oración son los nodos, cada nodo se evalúa de acuerdo con una técnica de ponderación, con esto, las abejas encuentran el mejor camino de solución, es decir, la secuencia más probable de etiquetas. Esta propuesta se evalúa utilizando el corpus KALIMAT [55], donde el etiquetador propuesto logró mejores valores de precisión en contraste con: a) enfoques tradicionales como los modelos ocultos de Markov y basados en reglas y b) métodos híbridos basados en reglas de Al-Taani y Al-Rub [56] en combinación con modelos ocultos de Markov.

En 2017, Sierra *et al.* [25] presentan un etiquetador de partes del discurso basado en Global-Best Harmony Search (GBHS) que incluye conocimiento del problema mediante una estrategia de optimización local basada en el algoritmo Hill Climbing [32]. En los experimentos se utilizó el corpus Brown del Inglés [20] con validación cruzada de 5 folders usando el conjunto de etiquetas Universal [57]. En 2018, Sierra *et al.* [26] presentan la construcción de un corpus etiquetado para el Nasa Yuwe y la aplicación

de algunos algoritmos de etiquetado, a saber: el memético propuesto en [25] basado en GBHS, un etiquetador aleatorio y tres versiones de HSTAGger[8]. Para los experimentos utilizaron validación cruzada de 10 folders y la estrategia Leave-One-Out, donde el algoritmo memético obtuvo los mejores valores de precisión comparado con los otros etiquetadores.

En 2010, Forsati *et al.* [58] presentaron un algoritmo para el problema de etiquetado basado en la metaheurística HS, el etiquetador es llamado Harmony Search Tagger (HSTAG) y se evaluó sobre el corpus Brown usando un conjunto de 47 etiquetas. El HSTAG fue comparado con los etiquetadores basados en algoritmos genéticos y recocido simulado presentados en [59], donde HSTAG obtuvo mejores valores de precisión. En 2012, Forsati y Shamsfard [38], introducen la cooperación entre los algoritmos evolutivos y los enfoques basados en estadísticas para el etiquetado, obteniendo un etiquetador llamado BEETAGger, basado en BCO, la función fitness fue simplificada en gran medida recurriendo a modelos estadísticos. El entrenamiento y evaluación, utilizó el corpus Brown [20], mostrando superioridad de BEETAGger en comparación con los otros etiquetadores utilizados. Luego en 2014, Forsati y Shamsfard [60], realizaron una comparación entre el desempeño de estrategias evolutivas como BCO y HS con los etiquetadores HSTAGger y BEETAGger propuestos anteriormente, teniendo en cuenta modificaciones en el tamaño de los datasets y reducción del conjunto de etiquetas para el desarrollo de los experimentos. En 2015, Forsati y Shamsfard [24], presentaron dos mejoras del etiquetador HSTAger, llamados HSTAger(I) y HSTAger(II), que incrementan la eficiencia de búsqueda y mejoran la selección de nuevas soluciones para la memoria armónica. Para el entrenamiento y evaluación, usaron tres data sets del idioma Inglés, los dos primeros del corpus Brown [20] y el último de una sección del corpus PennTreenbank [22], el tamaño del conjunto de etiquetas fue de 47 y los resultados mostraron que el HSTAGer(II) supera a los otros etiquetadores HSTAGer y HSTAGer(I).

En 2012, Silva *et al.* [61], modelaron el problema de etiquetado como un problema de optimización combinatoria, usando un algoritmo de clasificación para evolucionar un conjunto de reglas de desambiguación, luego usan un algoritmo genético para etiquetar las palabras de una oración (el cual es llamado GA-Tagger) usando las reglas encontradas anteriormente. El etiquetador se desarrolló utilizando los recursos disponibles en el paquete NLTK (Natural Language Toolkit) sobre el corpus Brown [20] y el 10% del corpus PennTreebank [22]. Para el conjunto de etiquetas se activó la opción de etiquetas simplificadas de NLTK. En 2013, Silva *et al.* [21] presentaron la

[8] HSTagger, HSTagger3 y HSTagger3.

aplicación del etiquetador PSO-Tagger dividiendo el etiquetado en dos tareas como en [61], las pruebas se hicieron para Inglés con el corpus Brown [20] y un 10% del corpus PennTreenbank [22], y para el portugués sobre el corpus Mac-Morpho [23] con un conjunto de 22 etiquetas, los resultados mostraron una mejora en los valores de precisión en comparación con [61]. También en 2013, Silva et al. [62] dividen el problema en dos tareas: una tarea de aprendizaje y una tarea de optimización, que ayuda al descubrimiento de reglas usando la computación evolutiva, para ello, utilizaron dos algoritmos de búsqueda diferentes, un algoritmo genético (GA-Tagger) y un PSO binario (PSO-Tagger). El entrenamiento y evaluación se hicieron utilizando el corpus Brown [19] del Inglés, mostrando que la mejor precisión fue lograda por PSO-Tagger.

En 2013, Ekbal y Saha [2], abordan el problema de etiquetado usando optimización mono-objetivo y multiobjetivo (basado en Simulated Annealing-Based Multiobjective Optimization Algorithm [63]) explotando la capacidad de búsqueda del algoritmo recocido simulado. Las pruebas fueron realizadas sobre dos lenguas indias (bengali e hindi) obteniendo mejores valores de precisión con la optimización multiobjetivo.

2.2.2. Corpus etiquetados para la identificación de partes del discurso

En 2018, Sierra et al. [26] presentaron una corpus para Nasa Yuwe, que cuenta con 175 frases etiquetadas, con longitud máxima de 34 palabras y mínima de 1 y consta de 1176 palabras, con una longitud máxima de 19 caracteres y mínima de 1 carácter, con un promedio 7.5 caracteres. Fue etiquetado manualmente y tiene dos versiones: una etiquetada con el conjunto de etiquetas propuesto por [64] y la otra se alineó con el conjunto de etiquetas universales [57].

En 2016, Alonso y Zeman [65], presentan el UD Spanish Ancora, el cual contiene 17680 oraciones y 547682 tokens. Fue formado a partir del corpus Ancora, pero se hace la conversión del conjunto de etiquetas a dependencias Universales [66], el cual consta de 17 etiquetas.

En 2015, Lavid *et al.* [67], presentan un corpus paralelo de alta calidad para el par de idiomas Inglés – Español, el corpus se compone de textos originales en los dos idiomas y textos traducidos en ambas direcciones, el corpus puede ser utilizado en una serie de disciplinas como: estudios lingüísticos, estudios de traducción, máquinas de traducción, traducción asistida por computadora, aprendizaje de idiomas asistido por computadora y extracción de terminología

En 2013, El-Haj y Koulali [55] presentaron el corpus KALIMAT el cual fue creado automáticamente para ayudar en las diferentes áreas del procesamiento del lenguaje

natural árabe. En el proceso de creación del corpus fue recogido una colección de datos de 20291 artículos, y para anotar la colección de datos fue usado el etiquetador Stanford POSTagger [68], que utiliza un conjunto de 36 etiquetas basado en la codificación del proyecto PennTreenbank. Algunas de estas etiquetas son: sustantivo (NN), sustantivo plural (NNS), nombre propio (NNP), verbo (VB) y Adjetivo (JJ).

En 2012, el Institut Universitari de Lingüística Aplicada (IULA) [69] presentó el corpus IULA Spanish LSP Treebank que consta de 40000 oraciones de la lengua castellana, tomadas del IULA technical corpus. Las oraciones fueron automáticamente anotadas con información POS, y semiautomáticamente con información sintáctica. Las etiquetas se basan en el conjunto de etiquetas EAGLES [70] para el español, el cual contiene 577 etiquetas.

En 1993, Marcus et al. [22] presentaron el corpus PennTreebank, que consta de 4.5 millones de palabras del Inglés americano. Este corpus ha sido anotado con información de partes del discurso, el conjunto de etiquetas está conformado por 36 etiquetas de POS y otras 12 etiquetas de puntuación y símbolos de moneda.

En 1979, Francis y Kucera [20] presentaron el corpus Brown, que es una recopilación de 500 archivos de texto plano en Inglés. El conjunto de etiquetas está conformado por 87 categorías, las cuales se subdividen, obteniendo un total de 472 etiquetas. El corpus Brown es ampliamente utilizado en los diferentes trabajos de etiquetado revisados.

2.2.3. Conjunto de etiquetas

En 2017, Zeroual *et al.* [71] diseñaron un conjunto de etiquetas para la lengua arábica, que consiste en establecer niveles jerárquicos de las categorías de las palabras, estos niveles jerárquicos permiten una expansión más fácil y mejoran el etiquetado. El conjunto de etiquetas está constituido por tres categorías principales: sustantivo, verbo y partícula, que a su vez se dividen en más subcategorías, obteniendo un total de 110 etiquetas. En comparación con el conjunto de etiquetas arábico clásico, la propuesta logra mejores resultados para el etiquetado, probado en un etiquetador basado en estadística llamado TreeTagger [72].

En 2016, Kabashi y Proisl [73], presentaron un conjunto de etiquetas de grano fino para el lenguaje albanés, el cual sigue un enfoque tradicional para las categorías de palabras. Estas categorías son: sustantivos, verbos, adjetivos, adverbios, pronombres, preposiciones, conjunciones, numerales, partículas e interjecciones. Después, en cada categoría se añade información morfosintáctica para proponer nuevas etiquetas, y así obtener un total de 67 etiquetas.

En 2014, más de 200 colaboradores presentaron un proyecto abierto llamado dependencias universales [66], el cual es un conjunto de anotación para muchos lenguajes, que está basado en las dependencias de Stanford [74], etiquetas universales [57], y el interlingua Interset para los conjuntos de etiquetas morfosintácticas [75], dando como resultado 17 categorías de etiquetas, y dependiendo de las necesidades de un lenguaje, permite extensiones de estas categorías.

En 2012, Petrov *et al.* [57], presentan un conjunto de 12 etiquetas[9] comunes para 25 lenguajes (a partir de 25 conjuntos de etiquetas) provenientes de trabajos previos buscando mejorar la precisión de los etiquetadores a través de varios lenguajes. Esta propuesta ha sido muy aceptada y utilizada en diferentes trabajos de identificación de partes del discurso [76]. Para analizar y comparar el desempeño del conjunto de etiquetas, se utilizaron los corpus de etiquetado Brown [20] y PennTreebank [22].

En la **Figura 2** se presentan los dos principales enfoques y algunos de sus algoritmos aplicados bajo un determinado corpus.

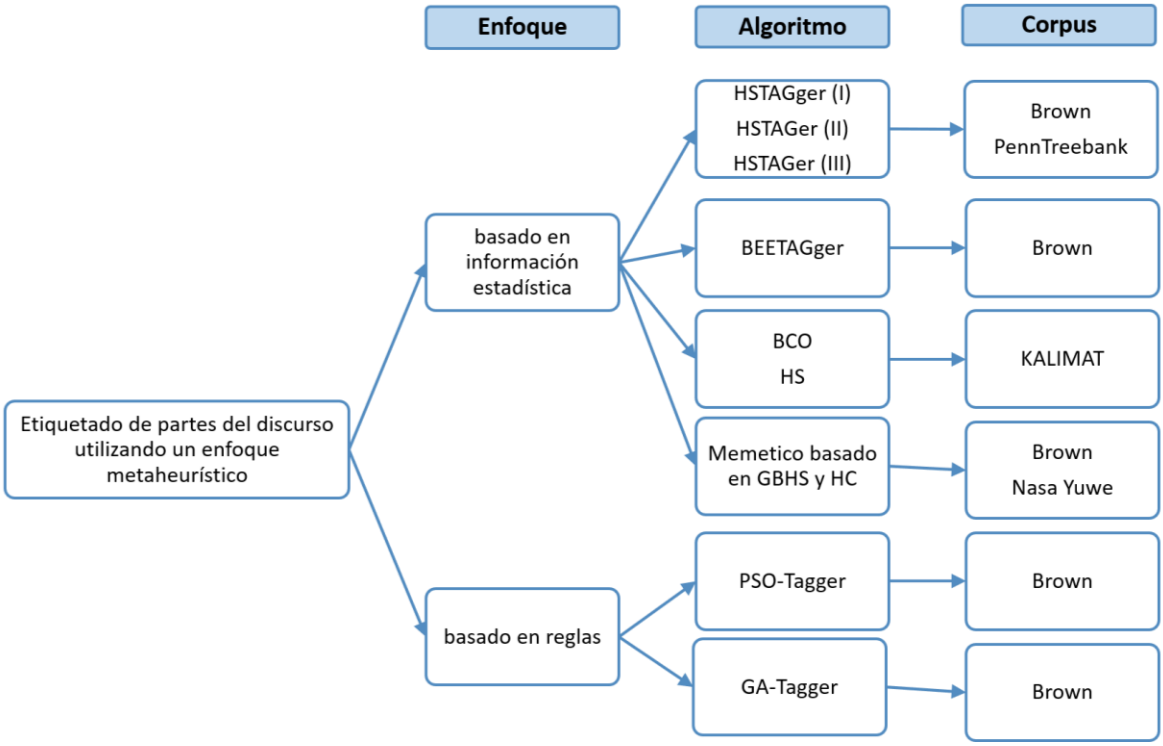


Figura 2. Algoritmos en el etiquetado POST

[9] NOUN (sustantivos), VERB (verbos), ADJ (adjetivos), ADV (adverbios), PRON (pronombres), DET (determinantes y artículos), ADP (preposición y posposición), NUM (numerales), CONJ (conjunciones), PRT (partículas), '.' (puntuación) and X (Un atrapa-todo para otras categorías como abreviaturas o palabras extranjeras) [57].

De la revisión realizada se puede apreciar que:

- Los etiquetadores basados en algoritmos metaheurísticos obtienen resultados mejores o similares a los alcanzados con los enfoques tradicionales o híbridos, tanto en etiquetado de lenguas tradicionales como en lenguas no tradicionales.
- Solo existe una propuesta que utiliza un enfoque memético para el problema de etiquetado, esta presenta una sola versión memética, mejorando la versión metaheurística del algoritmo GBHS.
- La precisión obtenida por el algoritmo memético GBHS Tagger [25], que trabaja sobre el corpus etiquetado de Nasa Yuwe, no es un valor adecuado para implementar una herramienta software que se dedique al etiquetado.
- No todos los corpus revisados están mapeados al conjunto de etiquetas universal Petrov [57].
- Muchos de los corpus presentados tienen su propio conjunto de etiquetas, el cual incluye mucha información de grano fino.

Por lo tanto, en esta propuesta:

- Se implementó una base de datos global con los datasets Brown [25] para el Inglés, Nasa Yuwe [26] para el Nasa y para el Castellano se construyó un dataset con los datos del corpus IULA [69].
- Se consolidó una línea base para el castellano, donde se ejecutaron los algoritmos metaheurísticos que se presentan en [25].
- Se adaptaron los algoritmos metaheurísticos Jaya [27] y PSO [46] al problema de etiquetado, donde se procedió a convertir su espacio de soluciones continuo a discreto, además se refinaron los parámetros de cada algoritmo con experimentos sobre un conjunto de datos reducido. Después se hizo la ejecución de los algoritmos sobre los corpus completos de Inglés, Castellano y Nasa Yuwe.
- Se realizó una mejora del algoritmo memético GBHS2Tagger propuesto por [25], utilizando ventanas contextuales, obteniendo cuatro versiones del memético, las cuales fueron ejecutadas sobre un pequeño conjunto de datos del corpus castellano, determinando como la mejor versión la denominada GBHS2Pred2, posteriormente se ejecutó la mejor versión sobre el conjunto de datos completo, dando como resultado que la versión GBHS2Pred2 obtuvo mejores resultados de precisión sobre la línea base para el castellano.
- Se realizó la implementación del algoritmo metaheurístico Hill Climbing (HC) adaptado al problema de etiquetado, para su posterior inclusión en el mejor algoritmo metaheurístico, convirtiéndose en una nueva propuesta memética.

- Se adaptó el algoritmo metaheurístico Random Restart Hill Climbing (RRHC) al problema de etiquetado, el cual obtuvo muy buenos resultados, superando a algoritmos metaheurísticos poblacionales.
- Se presenta una nueva versión memética del mejor algoritmo metaheurístico que incluyo el algoritmo Hill Climbing como optimizador local.
- Se diseñó e implementó un servicio web que permite a un usuario seleccionar una frase y etiquetarla con los algoritmos metaheurísticos presentados en este trabajo, el cual se utiliza desde el prototipo software web desarrollado en este trabajo.

Capítulo 3

Adaptación de los algoritmos metaheurísticos al problema de etiquetado

En este capítulo, se presenta en primera instancia, la línea base utilizada para este proyecto, que involucra los algoritmos metaheurísticos seleccionados del estado del arte, los cuales fueron utilizados para hacer la comparación con los algoritmos implementados en este proyecto. En segunda instancia, se presenta la construcción de un dataset para la lengua castellana basado en el corpus IULA [69] y la ejecución de experimentos de los algoritmos de la línea base sobre este dataset. En tercera instancia, se presenta la adaptación de los algoritmos metaheurísticos Jaya [45] y PSO [46] al problema de etiquetado, con sus correspondientes experimentos para los corpus IULA [69], Brown [25] y Nasa Yuwe [26]. Finalmente, se presenta una síntesis de lo presentado en esta sección.

3.1 Línea Base Corpus Nasa Yuwe y Brown (inglés)

Para la conformación de los algoritmos de la línea base se tomaron los presentados en Sierra *et al.* [25], que obtuvieron muy buenos resultados en contraste con etiquetadores tradicionales y basados en metaheurísticas, así, los algoritmos de la línea base en este trabajo son las siguientes metaheurísticas:

El algoritmo memético GBHSTagger propuesto en [25], que está basado en Global Best Harmony Search GBHS como una estrategia de búsqueda global y Hill Climbing HC como una estrategia de búsqueda local; además de utilizar los mismos parámetros de su versión original (HMCR, PARMax, HMS y NI) también utiliza los parámetros: 1) probabilidad de optimización (ProbOpt), que se encarga de controlar el porcentaje de veces que se quiere aplicar el proceso de optimización local a la mejor solución (armonía) en la memoria de la armonía; 2) Número de vecinos (MaxNeighbours), el cual es evaluado en el proceso de optimización local; y 3) Alpha, que garantiza que los componentes de cada armonía en la población se generen de forma aleatoria (a partir de sus etiquetas posibles) o se tome la etiqueta con la probabilidad más alta. Todo esto permite al algoritmo obtener mejores resultados de precisión que su versión original. Las versiones de este algoritmo son: 1) GBHS Tagger, que no incluye el parámetro Alpha, es decir solo utiliza inicialización aleatoria para la población, además incluye el parámetro de probabilidad de optimización (ProbOpt). 2) GBHS Tagger 2, que incluye los parámetros Alpha y ProbOpt. 3) GBHS Tagger 3, que combina la

inicialización aleatoria y mejorada, se incluyó el parámetro de probabilidad de optimización (ProbOpt).

Los otros algoritmos metaheurísticos incluidos en la línea base son: 1) Azar [32], que etiqueta aleatoriamente las palabras de las oraciones; 2) HSTagger (HSTAGger [60]), HSTagger2 (HSTAger(I) [24]) y HSTagger3 (HSTAger(II) [24]).

En los experimentos realizados en [25] y [26] se pudo apreciar que, el algoritmo memético GBHS Tagger2 utilizando el optimizador local con probabilidades 0.3, 0.5 y 0.7 logró obtener los mejores resultados en comparación con otras metaheurísticas, todas evaluadas sobre los corpus Brown y Nasa Yuwe. El mejor valor de precisión para cada corpus se muestra en la **Tabla 1,** estos valores conforman la línea base de este trabajo para la lengua Inglés y Nasa Yuwe.

Tabla 1. Resultados para el algoritmo GBHS Tagger2 de [25] y [26]. Fuente: Propia

Corpus	Precisión
Nasa Yuwe [26]	66.5787
Brown [25]	95.1959

Revisando el estado del arte, no se evidencian trabajos relacionados al problema de etiquetado que utilicen algoritmos metaheurísticos evaluados sobre la lengua castellana, por esta razón se realizó la construcción de un data-set para el castellano y la ejecución de experimentos sobre éste mismo, con el fin de consolidar la línea base para esta lengua; en la siguiente sección, se describe el proceso realizado.

3.2 Línea Base Corpus IULA (Castellano)

3.2.1 Construcción de dataset IULA

El corpus IULA [69] original se encuentra almacenado en un archivo con formato CONLL, donde cada línea representa un sólo termino con una serie de 10 campos, separados por tabulaciones, alguno de estos campos es ID, Forma de la palabra, etiquetado POS de grano fino y de grano grueso, entre otros. Teniendo en cuenta este formato se realizó el procesamiento para configurarlo como un dataset adecuado para la configuración de los experimentos con los algoritmos POST. En primera instancia, se realizó un modelo de base de datos, que permitiera a cada algoritmo consultar de manera eficiente la información del corpus, como: términos, frases, conjunto de etiquetas, tablas estadísticas, procedimientos almacenados y el folder de la frase para hacer validación cruzada. La **Figura 3**, presenta el modelo de base de datos construido, que incluye las tablas y sus respectivas relaciones entre frases, términos, etiquetas propias de los términos del corpus y su correspondiente etiqueta Universal.

Tabla 2. Actividades definidas en cada ciclo. **Fuente:** Elaboración propia**.**

Ciclo	Observación	Identificación	Desarrollo	Pruebas
1	Revisión de metaheurísticas e investigación de corpus en castellano	Selección del corpus y mapeo de etiquetas al etiquetado universal	Diseño e implementación de la base de datos para el corpus IULA. Configuración y ejecución del experimento con el algoritmo memético propuesto en [25] y otros algoritmos metaheurísticos utilizados sobre el dataset IULA	Análisis y discusión de resultados de precisión obtenidos
2	Exploración de ventanas contextuales para el castellano.	Definición de los contextos implementados.	Propuesta e implementación de una mejora al mejor algoritmo del ciclo 1 basado en los contextos y ejecución de los experimentos y selección de la mejor versión preliminar del algoritmo.	Análisis de resultados y emisión de conclusiones

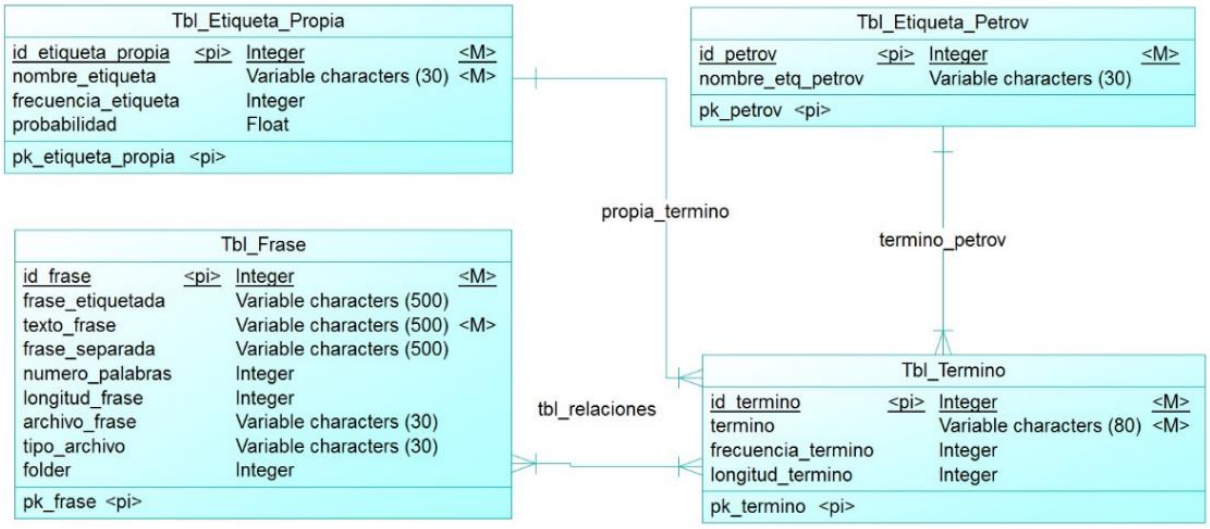


Figura 3. Modelo Conceptual de la Base de Datos

En segunda instancia, y partiendo del modelo conceptual presentado en la **Figura 3**, se generó el modelo físico de la base de datos para el motor SQL Server 2017. En [77] se encuentran los scripts de la base de datos que incluyen la estructura y los datos. En tercera instancia, se procedió a cargar la información en la base de datos construida mediante un algoritmo en Python, que incluye la lectura del archivo del Corpus IULA y toma el término con su etiqueta y hace la equivalencia de la etiqueta EAGLES [70] del término con su correspondiente etiqueta Universal [57] según el archivo de equivalencias “es-eagles.map” [78]. En la **Tabla 3** se presenta un fragmento de la

equivalencia entre etiquetas. Finalmente, con la lectura de estos términos se organizó cada frase hasta completar la totalidad de términos y frases del corpus en la base de datos.

Tabla 3. Equivalencia de etiquetas EAGLES a PETROV

Id	Etiq_IULA	Etiq_Petrov
1	AQ0MS0	ADJ
182	SPS00	ADP
184	RG	ADV
186	CC	CONJ
188	DD0MS0	DET
235	NCMS000	NOUN

Id	Etiq_IULA	Etiq_Petrov
259	Z	NUM
265	PP1CSN00	PRON
363	VMIP2P0	VERB
555	W	X
557	Fat	.

Como resultado del procesamiento del corpus, se logró consolidar una base de datos con 42.079 frases y 37.763 términos. Dada la cantidad de registros y la cantidad de consultas que debe atender, se llevó a cabo un proceso de indexación sobre las tablas, para las llaves con mayor número de relaciones.

Una vez fue completado el procesamiento del corpus IULA, se procedió a realizar los experimentos con los algoritmos GBHS Tagger, GBHS Tagger2, GBHS Tagger3, HSTagger, HSTagger2 y HSTagger3 y una caminata aleatoria (Azar), utilizando validación cruzada de cinco folders. Los resultados de estos experimentos se presentan en la sección 3.2.2 Experimentos con dataset IULA

Como resultado de los experimentos se pudo apreciar que el algoritmo memético GBHSTagger2, utilizando el optimizador local con probabilidades de 0.3, 0.5 y 0.7 logró obtener los mejores resultados para el corpus IULA, con una precisión de 97.6051%. Previamente en el Corpus Brown del inglés [20] en [25] obtuvo una precisión de 95.1959% y en Nasa Yuwe [26] una precisión de 66.5787%. Estos resultados dependen directamente del tamaño de su corpus y de las palabrabas desconocidas de cada uno de ellos, por ejemplo, en el caso del Nasa Yuwe se tiene un corpus de 175 frases, por lo que su resultado de precisión es el más bajo.

3.2.2 Experimentos con dataset IULA

Los experimentos fueron realizados sobre los algoritmos metaheurísticos: Azar (caminata aleatoria), GBHS Tagger, GBHS Tagger2, GBHS Tagger3, HSTagger, HSTagger2 y HSTagger3 presentados en [25]. Todos los algoritmos se ejecutaron sobre el dataset IULA de la sección 3.2.1 Construcción de dataset IULA.

Para la ejecución de los experimentos se utilizó un modelo cliente – servidor, con una base de datos en la nube (servidor), en donde los clientes (máquinas) solicitan cierto número de tareas, cada tarea tiene la frase y esta es ejecutada 30 veces en la máquina local, una vez finalizada la tarea se hace conexión con la base de datos en la nube y se graban los resultados, así hasta terminar las tareas solicitadas. Este proceso se repite por parte de los clientes hasta que se terminan las tareas por ejecutar en la base datos. Todos los experimentos se ejecutaron utilizando validación cruzada de 5 folders. En la **Tabla 4**, se muestra los conjuntos de datos (data sets) de prueba y entrenamiento para cada folder. Los parámetros usados para todos los algoritmos de la línea base fueron los definidos en [25].

Tabla 4. Conjunto de datos de prueba y entrenamiento para los experimentos – Corpus IULA. Fuente: Propia

Folder Prueba	Oraciones Prueba	Palabras Prueba	Folder Entrenamiento	Palabras Entrenamiento	Palabras comunes	Palabras desconocidas
1	8416	16316	2, 3, 4, 5	65521	12422	3894 (24%)
2	8416	16357	1, 3, 4, 5	65480	12494	3863 (24%)
3	8416	16466	1, 2, 4, 5	65371	12506	3960 (24%)
4	8416	16290	1, 2, 3, 5	65547	12409	3881 (24%)
5	8415	16408	1, 2, 3, 4	65429	12509	3899 (24%)

Tabla 5. Resultados de desempeño de los algoritmos en dataset IULA para Castellano. Fuente: Propia

Algoritmos	Parámetro (ProbOpt)	Número de oraciones	Precisión (%)	Desviación estándar	Precisión (%) Palabras desconocidas	Desviación estándar
Azar	-	42079	94,8647	6,7539	90,0717	7,9378
HSTagger 3	-	42079	95,3244	6,8978	89,2502	7,7518
HSTagger	-	42079	96,9699	5,8722	93,1611	7,5736
HSTagger 2	-	42079	96,9820	5,8321	93,1185	7,4774
GBHS Tagger 2	0	42079	97,1298	5,8183	93,4895	7,5877
GBHSTagger 3	0	42079	97,1580	5,8163	93,5645	7,6159
GBHSTagger	0	42079	97,1753	5,8456	93,6521	7,7179
GBHSTagger	0,3	42079	97,4002	5,8272	94,1479	7,8776
GBHSTagger	0,5	42079	97,4002	5,8272	94,1479	7,8776
GBHSTagger	0,7	42079	97,4002	5,8272	94,1479	7,8776
GBHSTagger 3	0,3	42079	97,4124	5,7942	94,1507	7,8123

Algoritmos	Parámetro (ProbOpt)	Número de oraciones	Precisión (%)	Desviación estándar	Precisión (%) Palabras desconocidas	Desviación estándar
GBHSTagger 3	0,5	42079	97,4124	5,7942	94,1507	7,8123
GBHSTagger 3	0,7	42079	97,4124	5,7942	94,1507	7,8123
GBHSTagger 2	0,3	42079	97,4306	5,7844	94,1928	7,8055
GBHSTagger 2	0,5	42079	97,4306	5,7844	94,1928	7,8055
GBHSTagger 2	**0,7**	**42079**	**97,4306**	**5,7844**	**94,1928**	**7,8055**

La muestra la precisión obtenida en orden ascendente y los valores de la desviación estándar para cada uno de los algoritmos, al mismo tiempo, que se puede apreciar que los mejores resultados son los del algoritmo GBHS Tagger 2. La medida de precisión usada para la evaluación de los algoritmos es la presentada en la **Ecuación 2**.

$$Precisión = \frac{Número\ Palabras\ correctamente\ etiquetadas}{Número\ de\ palabras\ de\ la\ frase}$$

Ecuación 2. Medida de Precisión. **Fuente:** [24]

A los resultados de los experimentos realizados se le aplicó la prueba estadística no paramétrica de Friedman, que confirma que el GBHS Tagger2 supera en rendimiento a los demás algoritmos. La **Tabla 6**, muestra los resultados obtenidos en la prueba. Esta prueba tiene 17 grados de libertad (82.319298) y un valor p de 1.9663E-10, que lo hace estadísticamente significativo.

Tabla 6. Resultados Test de Friedman. Fuente: Propia Uso KEEL [79]

Algoritmo	Ranking Friedman	Algoritmo	Ranking Friedman	Algoritmo	Ranking Friedman
GBHS2 0.3 0.5	1	GBHS3 0.7 0.5	3.6	GBHS3 0.0 0.5	11
GBHS2 0.5 0.5	1	GBHS 0.3	7.4	HSTagger2 0.5	13.2
GBHS2 0.7 0.5	1	GBHS 0.5	7.4	HSTagger	13.8
GBHS3 0.3 0.5	3.6	GBHS 0.7	7.4	HSTagger3 0.5	15.2
GBHS3 0.5 0.5	3.6	GBHS 0.0	10	Azar	15.8

3.3 Integración de corpus

Como parte de este trabajo se realizó la integración de los corpus (Brown, Nasa yuwe e IULA) utilizados en una sola base de datos, desarrollada en SQL Server. A continuación, en la **Figura 4**, se presenta el modelo conceptual de la base de datos diseñada para la integración de los tres corpus.

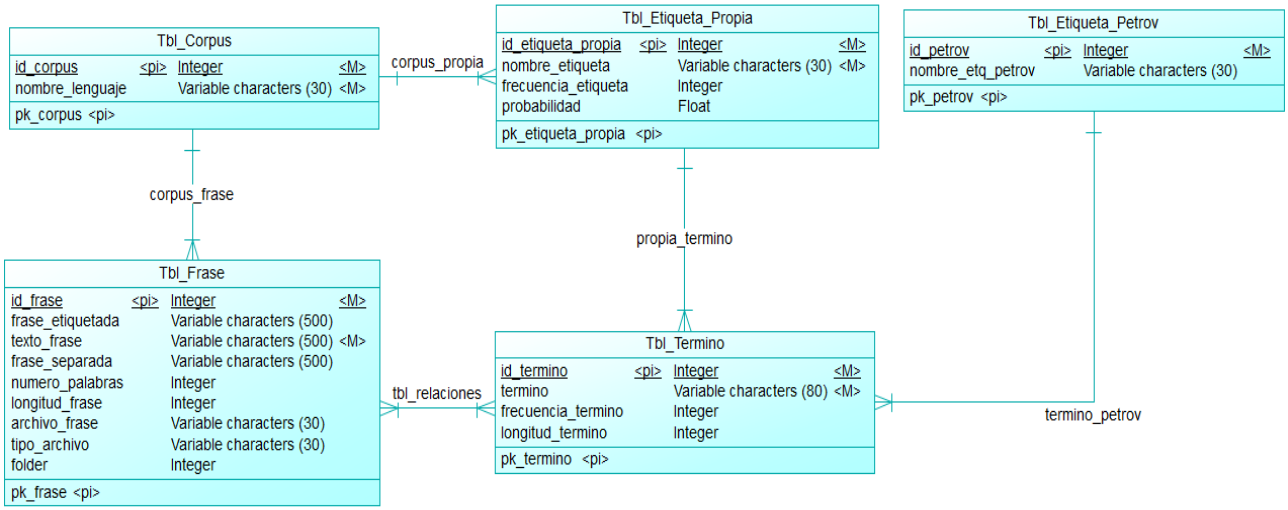


Figura 4. Modelo Conceptual de la Base de Datos Integradora. Fuente: Propia.

En el modelo conceptual presentado en la figura **Figura 4**, se destaca la tabla nombrada como “Tbl_Corpus”, la cual almacena los corpus utilizados en este trabajo, además la tabla se encuentra relacionada con la tabla nombrada como “Tbl_Frase” y así identificar las frases para cada corpus. Un aspecto a resaltar en este modelo es la relación de las tablas frase y términos (tbl_relaciones), la cual posteriormente se convertirá en una tabla que almacena la información de cada frase de los tres corpus, relacionada con los términos y el orden de cada termino en la frase, además el contexto a tener en cuenta para el término que se va a etiquetar, es decir, las etiquetas de sus predecesores, sucesores según la ventana a tener en cuenta.

En el anexo Digital 1 (shorturl.at/hjqrO), se encuentra el backup de la base de datos con los datos de los tres corpus.

3.4 Adaptación del algoritmo PSO al problema de etiquetado de partes del discurso

En la sección de trabajos relacionados, se pudo apreciar que el algoritmo PSO se ha utilizado en el problema de etiquetado desde un enfoque basado en reglas [62], donde obtuvo un buen desempeño, adicionalmente, revisando la literatura se evidencia que este algoritmo no ha sido utilizado desde el enfoque estadístico, por tanto, fue seleccionado en esta investigación para su adaptación al problema de etiquetado.

La adaptación del algoritmo PSO al problema de etiquetado se realizó de la siguiente manera: 1) se implementó el algoritmo PSO para problemas discretos, teniendo en

cuenta los parámetros originales $W, C1, C2$ y el parámetro de preservación de solución P; 2) la partícula se representa como la solución mostrada en la **Figura 5**.

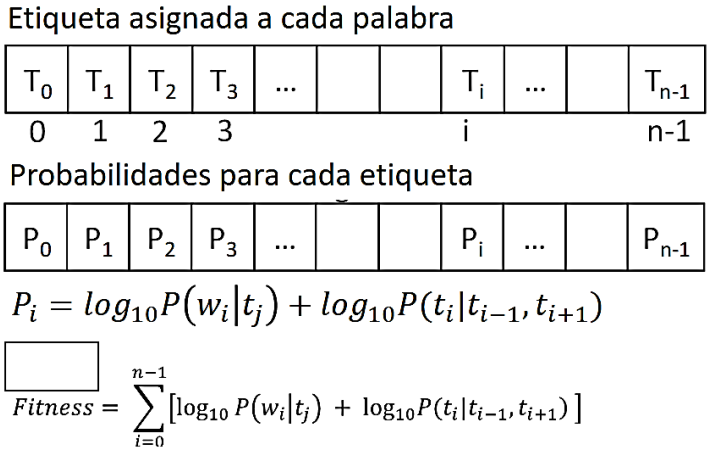


Figura 5. Representación de la solución.

A continuación, en el **Algoritmo 6,** se presenta el pseudocódigo de PSOTagger adaptado para el problema de etiquetado.

Algoritmo 6. PSOTagger

1. Definición de parámetros: $TamPoblacion$, MaxGeneraciones, W, C1, C2, P
2. Inicialización aleatoria de la $Población$
3. Definición de la mejor partícula global (G_{best})
4. Definición de P_{best} para cada partícula en el enjambre
5.
6. **para** $i = 1$ to MaxGeneraciones **hacer**
7. **para** $j = 1$ to $TamPoblacion$ **hacer** /* para cada partícula */
8. Evaluar la función fitness de la partícula (P_j) a través de la Ecuación 3
9. **si** (fitness (P_j) > fitness (P_{best}))
10. $P_{best} \leftarrow P_j$
11. **fin_si**
12. **fin_para**
13. Definir $G'_{best} \leftarrow Poblacion^{best}$ /* se define el mejor de la población actual*/
14. **si** (fitness (G'_{best}) > fitness (G_{best})) /* mejor actual es mayor a mejor anterior*/
15. $G_{best} \leftarrow G'_{best}$
16. **fin_si**
17. **para** $j = 1$ to TamPoblacion **hacer** /* para cada partícula */
18. **para** $k = 1$ to n **hacer** /* para cada palabra en la oración */
19. **si** $(Activo[j] = true)$ **entonces** /*¿la palabra tiene más de una etiqueta? */
20. $aleatorio \leftarrow r$ /* número aleatorio entre 0 y 1*/
21. **si** $(aleatorio < W)$ **entonces**
22. $P_k \leftarrow LB_k + r \times (UB_k - LB_k)$ /*selección de una etiqueta al azar */
23. **fin_si**
24. **sino** $(aleatorio < W + C1)$ **entonces**

Algoritmo 6. PSOTagger

```
25.                 P_k ← P_best(k) /* se selecciona la etiqueta del
                                       mejor histórico */
26.                 fin_si
27.                 sino (aleatorio < W + C1 + C2) entonces
28.                 P_k ← G_best(k) /* se selecciona la etiqueta del
                                       mejor Global */
29.                 fin_si
30.                 sino
31.                     P_k ← P_k /* se mantiene la etiqueta */
32.                 fin_si
33.             fin_si
34.         fin_para
35.     fin_para
36. fin_para
37. retornar G_best
```

Fuente: Propia

El **Algoritmo 6** presenta las siguientes características:

- En la línea 1, se definen seis parámetros, así:
  - El tamaño de la población (***TamPoblacion***).
  - El número máximo de generaciones (***MaxGeneraciones***).
  - El ***W*** que selecciona al azar una etiqueta para cada partícula (ver línea 21).
  - El ***C1*** selecciona la etiqueta del mejor histórico de cada partícula (ver línea 24).
  - El ***C2*** selecciona la etiqueta del mejor global del enjambre, para cada partícula (ver línea 27)
  - El ***P*** que mantiene los componentes de la partícula actual (ver línea 32)

- En la línea 2, se inicializa aleatoriamente la población
- En la línea 7 a la 12, se puede apreciar la actualización de la historia para cada partícula en el enjambre.
- En la línea 8 se realiza la evaluación de la función fitness, descrita en la **Ecuación 3**.

$$Fitness = \sum_{i=1}^{n} \left[\log_{10} P(w_i|t_j) + \log_{10} P(t_i|t_{i-1}, t_{i+1}) \right]$$

Ecuación 3. Función fitness. Fuente: adaptado de [25]

- En la línea 9 a la 11, se tiene la condición si: (fitness (P_j) > fitness (P_{best})), se actualiza el mejor histórico de la partícula en cuestión.
- En la línea 13, se define la mejor partícula de la población actual (G'best).

- En la línea 14 a la 16, se define el Mejor global (Gbest), según la condición: si (fitness(G'_{best}) > fitness (G_{best})), se actualiza la mejor partícula global.
- En la línea 17 a la 35, se hace el proceso de nueva generación para cada partícula:
  - En la línea 19 se tiene la condición: ($Activo[j] = true$), donde el vector $Activo$ contiene la validación de las palabras que tienen más de una posible etiqueta, si se cumple:
    - En la línea 20, se genera un número $aleatorio$ para cada palabra de la oración (partícula), y dependiendo de éste, con el fin de seleccionar uno de los siguientes criterios:
    - En la línea 21 a la 23**,** se tiene la condición**:** si ($aleatorio < W$) se selecciona una etiqueta al azar.
    - En la línea 24 a la 26**,** se tiene la condición**:** si ($aleatorio < W + C1$), se selecciona la etiqueta de la mejor historia de la partícula.
    - En la línea 27 a la 29**,** se tiene la condición**:** si ($aleatorio < W + C1 + C2$), se selecciona la etiqueta del mejor Global del enjambre.
    - En la línea 30 a la 32, se tiene la condición**:** si $aleatorio < W + C1 + C2 + P$, se conserva la etiqueta.
- En la línea 37, se retorna la frase etiquetada, que es la mejor partícula del enjambre.

Consolidado el algoritmo adaptado **PSOTagger**, se procede a realizar las pruebas con el ajuste de parámetros en cada corpus, buscando obtener una mejor versión del algoritmo.

3.3.1 Experimentos realizados para el corpus IULA

Los experimentos sobre el algoritmo PSOTagger fueron llevados a cabo inicialmente, sobre un data-set del corpus IULA de 5000 frases, presentado en la **Tabla 7,** utilizando validación cruzada de 5 folders.

Tabla 7. Dataset IULA de 5000 frases. Fuente: Propia.

Carpeta con los datos de prueba	Oraciones en los datos de prueba	Palabras en los datos de prueba	Carpeta con los datos de entrenamiento	Palabras en los datos de entrenamiento	Palabras comunes	Palabras desconocidas
1	1000	4322	2, 3, 4, 5	10662	2789	1533 (35%)
2	1000	4295	1, 3, 4, 5	10707	2807	1488 (35%)
3	1000	4245	1, 2, 4, 5	10687	2737	1508 (36%)
4	1000	4276	1, 2, 3, 5	10674	2755	1521 (36%)
5	1000	4360	1, 2, 3, 4	10616	2781	1579 (36%)

Teniendo en cuenta los parámetros del PSOTagger $W, C1, C2\ y\ P$, se procedió a hacer diferentes experimentos mediante un ajuste de parámetros con la técnica de prueba y error [80], en la **Tabla 8**, se pueden observar los experimentos que tuvieron resultados relevantes.

Tabla 8. Resultados de experimentos de 5000 frases para PSOTagger - Corpus IULA

#	Algoritmos	Frases	Precisión (%)	Desviación estándar	Justificación / Observaciones
1	PSOTagger 0.25 0.25 0.25	5000	92.571	7.906	Ninguna
2	PSOTagger 0.3 0.2 0.3	5000	92.672	7.924	Se incrementó el parámetro W, con respecto al anterior experimento, por consiguiente, aumento la precisión.
3	PSOTagger 0.3 0.2 0.4	5000	92.882	7.861	Se incrementó el parámetro C2, con respecto al anterior experimento, por consiguiente, aumento la precisión,
4	PSOTagger 0.2 0.2 0.5	5000	92.729	7.801	Se incrementó el parámetro C2, con respecto al anterior experimento, por consiguiente, disminuyo la precisión.
5	PSOTagger 0.4 0.2 0.3	5000	92.752	7.909	Se incrementó el parámetro W, con respecto al experimento 3, y por consiguiente disminuyo la precisión.
6	PSOTagger 0.3 0.3 0.3	5000	92.862	7.853	Se decremento el parámetro C2, con respecto al experimento 3, por consiguiente, disminuyo la precisión
7	PSOTagger 0.3 0.25 0.35	5000	92.849	7.848	Se decremento el parámetro C2, con respecto al experimento anterior, por consiguiente, disminuyo la precisión.
8	**PSOTagger 0.3 0.15 0.45**	**5000**	**92.883**	**7.821**	**Se aumentó el parámetro C2, con respecto al experimento 3, por consiguiente, aumentó la precisión.**
9	PSOTagger 0.3 0.12 0.48	5000	92.882	7.836	Se aumentó el parámetro C2, con respecto al experimento anterior, por consiguiente, disminuyó la precisión.

En la **Tabla 8** se muestra que el experimento 8, con parámetros $W = 0.3, C1 = 0.15, C2 = 0.45\ y\ P = 0.1$, sobrepasa en el valor de precisión a los demás

experimentos, por lo tanto, fue el que se ejecutó sobre el data-set completo del corpus IULA, presentado en la **Tabla 4**.

3.3.2 Experimentos realizados para el corpus Brown

Los experimentos sobre el algoritmo PSOTagger fueron aplicados a 1000 frases del corpus BROWN (presentado en la **Tabla 9**) seleccionadas al azar, utilizando validación cruzada de 5 folders.

Tabla 9. Data set completo del corpus Brown. Fuente: [25]

Carpeta con los datos de prueba	Oraciones en los datos de prueba	Palabras en los datos de prueba	Carpeta con los datos de entrenamiento	Palabras en los datos de entrenamiento	Palabras comunes	Palabras desconocidas
1	10595	23105	2,3,4,5	45113	18398	4707 (20.4%)
2	10600	22852	1,3,4,5	45199	18231	4621 (20.2%)
3	10600	23130	1,2,4,5	45009	18319	4811 (20.8%)
4	10600	22929	1,2,3,5	45130	18239	4690 (20.5%)
5	10603	23111	1,2,3,4	45025	18316	4795 (20.8%)

Teniendo en cuenta los parámetros del PSOTagger $W, C1, C2\ y\ P$, se procedió a realizar los mismos experimentos de la sección anterior, en la **Tabla 10**, se pueden observar los resultados de los experimentos llevados a cabo.

Tabla 10. Resultados de experimentos de 1000 frases para PSOTagger - Corpus BROWN. **Fuente:** Propia

#	Algoritmos	Frases	Precisión (%)	Desviación estándar
1	PSOTagger 0.0 0.3 0.15 0.45	1000	86.726	8.373
2	PSOTagger 0.3 0.3 0.15 0.45	1000	87.989	7.971
3	PSOTagger 0.5 0.25 0.25 0.25	1000	89.485	7.455
4	PSOTagger 0.5 0.3 0.15 0.4	1000	89.532	7.501
5	PSOTagger 0.5 0.3 0.2 0.25	1000	89.157	7.528
6	PSOTagger 0.5 0.3 0.2 0.4	1000	89.626	7.454
7	PSOTagger 0.5 0.3 0.3 0.2	1000	89.144	7.573
8	PSOTagger 0.5 0.3 0.3 0.3	1000	89.479	7.471
9	PSOTagger 0.5 0.35 0.15 0.4	1000	89.260	7.517
10	**PSOTagger 0.5 0.3 0.15 0.45**	**1000**	**89.650**	**7.448**

En la **Tabla 10** se muestra que el experimento 10, con inicialización mejorada, parámetros $W = 0.3, C1 = 0.15, C2 = 0.45\ y\ P = 0.1$, sobrepasa en el valor de precisión a los demás experimentos, por lo tanto, fue el que se ejecutó sobre el data-set completo del corpus Brown, presentado en la **Tabla 9.**

3.3.3 Experimentos realizados para el corpus Nasa Yuwe

Los experimentos sobre el algoritmo PSOTagger fueron aplicados a al corpus Nasa Yuwe (presentado en la **Tabla 11**), utilizando Leave-One-Out, o sea que utilizó una sola frase como prueba y las otras 174 frases como datos de entrenamiento (175 folders).

Tabla 11. Dataset Nasa Yuwe (Leave-One-Out). **Fuente:** [26]

Carpeta con los datos de prueba	Oraciones en los datos de prueba	Palabras en los datos de prueba	Carpeta con los datos de entrenamiento	Palabras en los datos de entrenamiento	Palabras comunes	Palabras desconocidas
1	1	3	2,3,...,175	1999	1	2
2	1	13	1,3,...,175	1989	7	6
80	1	34	1,...79,81,...,175	1968	17	17
174	1	21	1,2,...,173,175	1981	1	19
175	1	6	1,2,...,173,174	1996	1	5

Teniendo en cuenta los parámetros del PSOTagger, se procedió a realizar los mismos experimentos de la sección anterior, en la **Tabla 12** se pueden observar los resultados de los experimentos llevados a cabo.

Tabla 12. Resultados de experimentos de 175 frases para PSOTagger - Corpus Nasa Yuwe. Fuente: Propia

#	Algoritmos	Frases	Precisión (%)	Desviación estándar
1	PSOTagger 0,0 0,3 0,15 0,45	175	56,865	16,183
2	PSOTagger 0,3 0,3 0,15 0,45	175	56,407	15,708
3	PSOTagger 0,5 0,25 0,25 0,25	175	55,426	16,067
4	PSOTagger 0,5 0,3 0,15 0,4	175	55,829	15,941
5	PSOTagger 0,5 0,3 0,15 0,45	175	55,947	15,802
6	PSOTagger 0,5 0,3 0,2 0,25	175	54,814	16,061
7	**PSOTagger 0,5 0,3 0,2 0,4**	**175**	**55,936**	**16,052**
8	PSOTagger 0,5 0,3 0,3 0,2	175	54,557	16,006
9	PSOTagger 0,5 0,3 0,3 0,3	175	55,744	15,923
10	PSOTagger 0,5 0,35 0,15 0,4	175	55,799	15,835

En la **Tabla 12** se muestra que el experimento 7, con inicialización mejorada, parámetros $W = 0.3,\ C1 = 0.15,\ C2 = 0.45\ y\ P = 0.1$, sobrepasa en el valor de precisión a los demás experimentos; por lo tanto, el experimento 7 fue definido para su posterior comparación con los algoritmos metaheurísticos GBHS y PSO.

3.4 Adaptación del algoritmo metaheurístico Jaya al problema de etiquetado

El algoritmo Jaya es una nueva metaheurística propuesta por Rao [45], es un algoritmo novedoso, simple y eficiente para resolver problemas de optimización restringidos y no restringidos. En algunos trabajos donde se aplicó el algoritmo Jaya, los resultados muestran la superioridad de Jaya sobre otras metaheurísticas, como genetic algorithm (GA), PSO y civilized swarm optimization (CSO) [81].

Jaya se basa en el principio de que la solución de un problema, debe avanzar hacia la solución más conocida y alejarse de la peor solución. Los pasos involucrados en la aplicación del algoritmo Jaya se resumen de la siguiente manera:

- Inicializar el número de variables de diseño, tamaño de la población y criterio de terminación.
- Identificar la peor y la mejor solución en la población
- Compare la solución actualizada de la solución anterior, si la solución actualizada es mejor, reemplácela, de lo contrario, mantenga la anterior.
- Finalmente retorne la solución optima

El algoritmo Jaya en su versión original, ha sido desarrollado para resolver problemas de optimización con variables de diseño continuo [27], pero recientemente este algoritmo se ha implementado para resolver problemas de optimización de variables discretas y enteras [82]. Una estrategia de discretización es la presentada en [83], en donde, se consideran tres números aleatorios binarios (es decir, cada número toma 0 o 1) considerados para el propósito de selección. En la **Ecuación 4,** se presenta la función de actualización del algoritmo discreto DJaya.

$$X'_i(t+1) = b_1 X_i(\mathrm{t}) + b_2 X_B(\mathrm{t}) + b_3 X_W(\mathrm{t}), \qquad i = 1, 2, \dots, N_p$$

$$with\ b_1, b_2, b_3 \in \{0,1\} \text{ and } \sum_{j=1}^{3} b_j = 1$$

Ecuación 4. Discretización para el algoritmo Jaya**. Fuente:** [83]**.**

En la **Ecuación 4,** se puede ver que existe la posibilidad de que se seleccione uno y solo un elemento, dependiendo de los valores de los números aleatorios b1, b2 y b3 para una nueva solución, al mismo tiempo cuando una operación es seleccionada desde X_i, X_B o X_W debe verificarse que esta operación no se haya seleccionado, para asegurarse de que no se repita una solución.

La adaptación del algoritmo Jaya al problema de etiquetado se realizó de la siguiente manera: 1) se implementó el algoritmo DJaya para problemas discretos, teniendo en cuenta la discretización de [83]; 2) la solución se representa como la solución mostrada en la **Figura 5.** A continuación, en el **Algoritmo 7**, se presenta el pseudocódigo de JayaTagger adaptado al problema de etiquetado.

Algoritmo 7. JayaTagger

1. Definición de parámetros: $TamPoblacion$, MaxGeneraciones, $P4$
2. Inicialización aleatoria de la $Población$
3. Definición de la mejor solución en la población (X_B)
4. Definición de la peor solución en la población (X_W)
5. **para** $i = 1$ to MaxGeneraciones **hacer**
6. **para** $j = 1$ to $TamPoblacion$ **hacer** /* para cada partícula */
7. X_{new} /* generar una nueva solución*/
8. **para** $h = 1$ to n **hacer** /* para cada palabra en la oración */
9. **si** ($Activo[h] = true$) **entonces** /* ¿la palabra actual tiene más de una posible etiqueta? */
10. $aleatorio \leftarrow r(1, P4)$
11. **según** ($aleatorio$)
12. **caso 1**: /*preserve la etiqueta de la solución*/
13. $X_{new(h)} = P_j(h)$
14. **caso 2**: /*Seleccione la etiqueta de la mejor solución*/
15. $X_{new(h)} = X_B(h)$
16. **caso 3**: /*Seleccione la etiqueta aleatoriamente*/
17. $X_{new(h)} = LB_h + r \times (UB_h - LB_h)$
18. **caso 4**: /*Seleccione la etiqueta de la peor solución*/
19. $X_{new(h)} = X_W(h)$
20. **fin_si**
21. **fin_para**
22. Evaluar la función fitness de la nueva solución (X_{new}) (**Figura 5**)
23. **si** (fitness (X_{new}) > fitness (P_j))
24. $P_j \leftarrow X_{new}$
25. **fin_si**
26. **fin_para**
27. **fin_para**
28. **retornar** X_B **/***Retorna solución de la población*/

Fuente: Propia

Este algoritmo presenta las siguientes características:

- Los únicos parámetros definidos para el algoritmo son: tamaño de la población ($TamPoblacion$), Criterio de parada (MaxGeneraciones) y el parámetro cuatro ($P4$), el cual controla que la nueva solución seleccione una etiqueta de la peor solución (X_W).

- Para escoger las etiquetas de la nueva solución, se genera un número entero aleatorio entre 1 y $P4$ (línea 10), según este número la nueva solución, solo seleccionara uno y solo un elemento de los operadores (línea 12 a la 19).

Consolidado el algoritmo adaptado **JayaTagger**, se procede a realizar las pruebas y el ajuste de parámetros para los corpus IULA, Brown y Nasa Yuwe, buscando obtener una mejor versión del algoritmo para cada lengua.

3.4.1 Experimentos realizados para el corpus IULA

Los experimentos sobre el algoritmo **JayaTagger** fueron llevados a cabo inicialmente, sobre el data-set del corpus IULA de 5000 frases, presentado en la **Tabla 7**, utilizando validación cruzada de 5 folders. La representación de la solución fue organizada como en la **Figura 5**.

En todos los experimentos se definió, el tamaño de población en 10 ($TamPoblacion$), Criterio de parada en 110 ($\mathrm{MaxGeneraciones}$). El parámetro p4 fue el único que cambio en cada experimento, debido a que si p4 = 4, la nueva solución, toma elementos de la peor solución y para nuestro problema, donde se tiene muy pocas opciones para cada etiqueta no es conveniente hacerlo.

Teniendo en cuenta lo anterior, se procedió hacer diferentes experimentos mediante un ajuste de parámetros con la técnica de prueba y error. En la **Tabla 13** se pueden observar los resultados de los experimentos relevantes sobre el corpus IULA.

Tabla 13. Resultados de experimentos de 5000 frases para JayaTagger – Corpus IULA. Fuente: Propia.

#	Algoritmos	P4	Frases	Precisión (%)	Desviación estándar	Justificación / Observaciones
1	**JayaTagger 0.0**	**4**	**5000**	**92.965**	**7.841**	**Se inició con una inicialización aleatoria y que incluyera todos los operadores.**
2	JayaTagger 0.0	3	5000	90.737	8.351	Se siguió con inicialización aleatoria, y no se incluyó el operador de peor solución.
3	JayaTagger 0.5	4	5000	92.918	7.746	Se inicia con inicialización mejorada, y se incluye todos los operadores.
4	JayaTagger 0.5	3	5000	88.727	9.732	Se siguió con inicialización mejorada, y no se incluyó el operador de peor solución

En la **Tabla 13**, se muestra que el experimento 1, con inicialización aleatoria y la inclusión de todos los operadores (p4 = 4), sobrepasa en el valor de precisión a los

demás experimentos; por lo tanto, fue el que se ejecutó sobre el data-set completo del corpus IULA, presentado en la **Tabla 4**.

3.4.2 Experimentos realizados para el corpus Brown

los experimentos sobre el algoritmo JayaTagger, fueron llevados a cabo sobre las mismas frases seleccionadas en la ejecución del algoritmo PSOTagger sobre el corpus Brown de la sección 3.3.2 Experimentos realizados para el corpus Brown. Los experimentos se realizaron utilizando validación cruzada de 5 folders.

Para la ejecución de JayaTagger sobre el corpus Brown, se procedió a realizar los mismos experimentos presentados en la sección anterior. En la **Tabla 14**, se pueden observar los resultados de los experimentos.

Tabla 14. Resultados de experimentos de 1000 frases para JayaTagger – Corpus Brown. Fuente: Propia.

#	Algoritmos	P4	Frases	Precisión (%)	Desviación estándar
1	JayaTagger 0.0	4	1000	87.819	8.106
2	JayaTagger 0.0	3	1000	86.433	7.999
3	JayaTagger 0.5	4	1000	90.567	7.297
4	**JayaTagger 0.5**	**3**	**1000**	**92.916**	**6.601**

En la **Tabla 14** se muestra que el experimento 4, con inicialización mejorada y la exclusión del operador de peor solución (p4 = 3), sobrepasa en el valor de precisión a los demás experimentos; por lo tanto, fue el que se ejecutó sobre el data-set completo del corpus Brown, presentado en la **Tabla 14**.

3.4.3 Experimentos realizados para el Corpus Nasa Yuwe

Los experimentos sobre el corpus Nasa Yuwe, para el algoritmo JayaTagger fueron llevados a cabo utilizando Leave-One-Out, es decir, se utilizó una sola frase como prueba y las otras 174 frases como datos de entrenamiento (175 folders). El conjunto de datos y entrenamientos es el presentado en la **Tabla 11**.

Para la ejecución de JayaTagger sobre el corpus Nasa Yuwe, se procedió a realizar los mismos experimentos presentados en la sección anterior. En la **Tabla 15**, se pueden observar los resultados de los experimentos.

Tabla 15. Resultados de experimentos para JayaTagger – Corpus Nasa Yuwe. Fuente: Propia

#	Algoritmos	P4	Frases	Precisión (%)	Desviación estándar
1	**JayaTagger 0.0**	**4**	**175**	**57.084**	**15.977**
2	JayaTagger 0.0	3	175	54.435	14.903
3	JayaTagger 0.5	4	175	56.325	15.901
4	JayaTagger 0.5	3	175	48.884	16.805

.

En la **Tabla 15** se muestra que el experimento 1, con inicialización aleatoria y la inclusión de todos los operadores (p4 = 4), sobrepasa en el valor de precisión a los demás experimentos.

3.5 Resultados de los algoritmos metaheurísticos adaptados al problema de etiquetado.

En esta sección se presenta los resultados de la ejecución de los etiquetadores PSOTagger, JayaTagger y GBHS Tagger2 sobre los datasets completos de los corpus IULA, Brown y Nasa Yuwe, en pro de seleccionar el mejor algoritmo, basado en la medida de precisión.

En el anexo digital 3 (shorturl.at/cdgO3), se presenta un documento con un poco más de detalle sobre la configuración para realizar los experimentos.

3.5.1 Resultados del mejor algoritmo metaheurístico para el corpus IULA

Los experimentos para el corpus IULA se ejecutaron con los algoritmos metaheurísticos PSOTagger, JayaTagger y GBHS Tagger2, utilizando el data-set completo, presentado en la **Tabla 4**; cada experimento fue configurado de la siguiente manera:

- **PSOTagger:** se tomó el experimento número 8 de la **Tabla 8** con parámetros $W = 0.3,\ C1 = 0.15,\ C2 = 0.45\ y\ P = 0.1$, que fue el que superó en precisión a los demás experimentos, para su ejecución se utilizó validación cruzada de 5 folders.

- **JayaTagger:** se tomó el experimento número 1 de la **Tabla 13**, con inicialización aleatoria y la inclusión de todos los operadores (p4 = 4), el cual superó en precisión a los demás experimentos, para su ejecución se utilizó validación cruzada de 5 folders.

- **GBHSTagger2:** se tomó el resultado de la **Tabla 5**, en donde se utilizó validación cruzada de 5 folders y el optimizador local con probabilidades 0.0, 0.3, 0.5 y 0.7, el cual superó en precisión a las demás metaheurísticas.

En la **Tabla 16**, se presentan los resultados de los experimentos realizados para cada algoritmo.

Tabla 16. Resultados de experimentos para el Corpus IULA. Fuente: Propia.

Algoritmo	Frases	Precisión	Desviación estándar
PSOTagger	42079	96,7738	5,6077
JayaTagger	42079	96,8592	5,5963
GBHSTagger2	**42079**	**97,4306**	**5,7844**

En la **Tabla 16**, se puede observar que el algoritmo GBHS Tagger2 superó a los algoritmos JayaTagger y PSOTagger en valor de precisión.

3.5.1 Resultados del mejor algoritmo metaheurístico para el corpus Brown

Los experimentos para el corpus Brown se ejecutaron con los algoritmos metaheurísticos PSOTagger, JayaTagger y GBHSTagger, utilizando el data-set completo, presentado en la **Tabla 9**; cada experimento fue configurado de la siguiente manera:

- **PSOTagger:** se tomó el experimento número 10 de la **Tabla 10**, con parámetros $W = 0.3,\ C1 = 0.15,\ C2 = 0.45\ y\ P = 0.1$, que fue el que superó en precisión a los demás experimentos, para su ejecución se utilizó validación cruzada de 5 folders.

- **JayaTagger:** se tomó el experimento número 4 de la **Tabla 14**, con inicialización aleatoria y la inclusión del operador de peor solución (p4 = 3), el cual superó en precisión a los demás experimentos, para su ejecución se utilizó validación cruzada de 5 folders.

- **GBHSTagger:** se replicó el experimento hecho en [25], bajo las mismas condiciones, es decir, la misma semilla, con los mismos ordenadores y utilizando validación cruzada de 5 folders.

En la **Tabla 17**, se presentan los resultados de los experimentos realizados para cada algoritmo.

Tabla 17. Resultados de experimentos para el Corpus Brown. Fuente: Propia

Algoritmo	Frases	Precisión	Desviación estándar
PSOTagger	53100	89,6933	7,0810
JayaTagger	53100	93,0519	6,3054
GBHS Tagger2	**53100**	**94.8232**	**6.3292**

En la **Tabla 17**, se puede observar que el algoritmo GBHSTagger2 superó a los algoritmos JayaTagger y PSOTagger en valor de precisión.

3.5.3 Resultados del mejor algoritmo metaheurístico para el corpus Nasa Yuwe

Para la comparación del corpus Nasa Yuwe, se seleccionó el mejor experimento para cada algoritmo metaheurístico, así:

- **PSOTagger:** se tomó el resultado del experimento 7, presentado en la **Tabla 12** con parámetros $W = 0.3,\ C1 = 0.2,\ C2 = 0.4\ y\ P = 0.1$, que fue el que superó en precisión a los demás experimentos.

- **JayaTagger:** se tomó el resultado del experimento número 1 de la **Tabla 15**, con inicialización aleatoria y la inclusión de todos los operadores (p4 = 4), el cual superó en precisión a los demás experimentos.

- **GBHSTagger2:** se replicó el experimento hecho en [26], bajo las mismas condiciones, es decir, la misma semilla, con los mismos ordenadores y utilizando la técnica Leave-One-Out.

En la **Tabla 18**, se presentan los resultados de los experimentos realizados para cada algoritmo.

Tabla 18. Resultados de experimentos para el Corpus Nasa Yuwe. Fuente: Propia.

Algoritmo	Frases	Precisión	Desviación estándar
PSOTagger	175	55,936	16,052
JayaTagger	175	57.084	15.977
GBHSTagger2	**175**	**60,0749**	**16.7585**

En la **Tabla 18**, se puede observar que el algoritmo GBHSTagger2 superó a los algoritmos JayaTagger y PSOTagger en valor de precisión para el corpus Nasa Yuwe.

En el anexo Digital 2 (shorturl.at/mBDHU), se encuentran los algoritmos adaptados al problema de etiquetado y presentados en esta sección.

3.6 Síntesis

En el presente capítulo, se realizó la construcción de un dataset del corpus IULA para la lengua castellana, el cual obtuvo muy buenos resultados, en comparación con los resultados de la línea base de Inglés y Nasa Yuwe. Adicionalmente, se realizó la integración de los tres corpus en un base de datos unificada en SQL Server.

Sumado a lo anterior, en esta sección también se presentó, la adaptación de los algoritmos metaheurísticos PSO y Jaya al problema de etiquetado, que fueron evaluados sobre los corpus Brown, IULA y Nasa Yuwe, obteniendo muy buenos antecedentes al problema de etiquetado para las lenguas respectivas. Sin embargo, en la comparación de los etiquetadores se observó que el etiquetador GBHS2Tagger superó a PSOTagger y JayaTagger, por lo tanto, este algoritmo fue seleccionado para realizar una nueva versión memética, en busca de obtener mejores resultados que los presentados hasta el momento. En el Anexo Digital 2 (shorturl.at/mBDHU), se encuentran los algoritmos adaptados e implementados en este trabajo. En el capítulo 4, se presenta la nueva versión memética del GBHS, además de una mejora al etiquetador GBHS2 Tagger2 para el corpus Castellano y por último un algoritmo metaheurístico de estado simple que obtiene muy buenos resultados en el etiquetado.

Capítulo 4

Propuesta de un algoritmo memético para el problema de etiquetado

En este capítulo se presenta: la adaptación al problema de etiquetado de los algoritmos metaheurísticos Hill Climbing (HC) y Random Restart Hill Climbing (RRHC); la inclusión de un optimizador local al algoritmo GBHS, obteniendo una nueva versión memética; y, una mejora al algoritmo memético GBHS2Tagger evaluado, sobre el corpus IULA, que mejora la precisión en el etiquetado.

4.1 Adaptación del algoritmo metaheurístico Hill Climbing al problema de etiquetado

El algoritmo Hill Climbing, HC, pertenece a las metaheurísticas de estado simple, como se pudo observar en la sección 2.1.2 Algoritmos metaheurísticos y meméticos, por tanto, solo se maneja una única solución, que se mueve iterativamente a una mejor solución vecina, hasta llegar a un óptimo local [84]. El HC original está diseñado para problemas de optimización continua, teniendo en cuenta esto se hizo el proceso de adaptación del algoritmo HC al problema de etiquetado, de la siguiente manera: 1) se utilizó una única solución donde se guarda las etiquetas de la oración a etiquetar; 2) la solución se representa como la solución mostrada en la **Figura 5;** 3) se utilizó una estrategia que selecciona entre dos métodos la palabra a etiquetar, el primer método, selecciona la palabra con la peor etiqueta en la solución y el segundo método, selecciona una palabra al azar sin importar la condición, evitando así, quedar atrapado en óptimos locales.

Teniendo en cuenta las consideraciones descritas, en el **Algoritmo 8** se presenta el algoritmo HC adaptado al problema de etiquetado.

Algoritmo 8. HCTagger

1. Definición del parámetro $Prob$(Probabilidad para seleccionar la estrategia de selección de la palabra a etiquetar)
2. Definición del parámetro N (Número máximo de evaluaciones de la función objetivo)
3. Inicialización aleatoria de la $Solucion$
4. ListaPosibilidadesEtiquetas /* Esta lista contiene todas las posibilidades de las palabras a etiquetar*/
5. **para** $i = 1$ to N **hacer**

Algoritmo 8. HCTagger

6. $aleatorio \leftarrow r(0,1)$ /* Se genera un numero aleatorio entre 0 y 1*/
7. $X_{new} \leftarrow$ Copiar($Solucion$)
8. **si** ($aleatorio \leq Prob$) **entonces** /*Si el número aleatorio es menor que $Prob$ definido en la configuración del experimento*/
9. $pos \leftarrow$ r(etiquetas) /* Se selecciona la posición de la palabra al azar */
10. **sino**
11. $pos \leftarrow$ indice de la palabra con la probabilidad mas baja /* Se selecciona la posición de la palabra basado en la probabilidad más baja
12. **fin_si**
13. $j \leftarrow LB_{pos} + r \times (UB_{pos} - LB_{pos})$ /*Se selecciona un índice al azar de las posibilidades de la palabra selecciona (pos)
14. $X_{new(pos)} =$ ListaPosibilidadesEtiquetas[pos][j] /*Se asigna la etiqueta a la nueva solución */
15. **si** (fitness (X_{new}) > fitness ($Solucion$)) **entonces**
16. $Solucion \leftarrow X_{new}$ /* Si la nueva solución supera a la actual se reemplaza */
17. **sino**
18. ListaPosibilidadesEtiquetas(pos).eliminar(j) **/***sino se elimina la posibilidad de la lista**/**
19. **fin_si**
20. **fin_para**
21. **retornar** $Solucion$

Fuente: Propia

Este algoritmo presenta las siguientes características:

- El parámetro $Prob$ define la probabilidad de selección de la nueva palabra a etiquetar, por un lado, escoge la palabra de manera aleatoria y por otro, selecciona la palabra con la probabilidad más baja (línea 8 a la 12).
- Se define la lista **ListaPosibilidadesEtiquetas,** la cual almacena las posibilidades de todas las palabras.
- Seleccionada la palabra a etiquetar, se escoge una etiqueta al azar de la lista **ListaPosibilidadesEtiquetas** y se asigna a la nueva solución (línea 13 a la 14).
- En caso de que la nueva solución supere a la solución actual, se reemplazara la solución actual, sino esta etiqueta se elimina de la **ListaPosibilidadesEtiquetas** para esa palabra (línea 15 a la 19).
- Al final de la ejecución del algoritmo se retorna la $Solucion$.

Consolidado el algoritmo adaptado **HCTagger**, se procede a realizar las pruebas y el ajuste de parámetros, en este caso solo se realizó para el corpus IULA, con el fin de consolidar el algoritmo, para luego, integrarlo al algoritmo GBHSTagger como optimizador local, y de ahí si realizar las respectivas pruebas para los otros corpus.

4.1.1 Experimentos realizados para el corpus IULA

Los experimentos sobre el algoritmo **HCTagger** fueron llevados a cabo inicialmente, sobre el data-set del corpus IULA de 5000 frases, presentado en la **Tabla 7**, utilizando validación cruzada de 5 folders. La representación de la solución fue organizada como en la **Figura 5**.

Los experimentos fueron definidos variando el parámetro $Prob$, con la técnica de prueba y error, en donde si un experimento superaba a otro, se tomaba como base para el siguiente experimento, así en la **Tabla 19** se muestra los resultados de los experimentos relevantes sobre el corpus IULA.

Tabla 19. Resultados de experimentos de 5000 frases para HCTagger – Corpus IULA. Fuente: Propia.

#	Algoritmos	Prob	Frases	Precisión (%)	Desviación estándar
1	HCTagger 0.0	0.5	5000	86.558	9.499
2	HCTagger 0.3	0.5	5000	87.118	9.228
3	**HCTagger 0.5**	**0.5**	**5000**	**87.543**	**9.078**

En la **Tabla 19** se muestra que el experimento 3, con inicialización mejorada y el parámetro $Prob = 0.5$ sobrepasa en el valor de precisión a los demás experimentos; por lo tanto, consolidado HCTagger, en la sección 4.4 Propuesta de un algoritmo memético para el problema de etiquetado, se procedió a integrarlo al algoritmo GBHS.

4.2 Adaptación del algoritmo metaheurístico Hill Climbing con reinicios aleatorios al problema de etiquetado

El algoritmo Random Restart Hill Climbing (RRHC) [42], es un conjunto de Hill Climbing, que parten desde estados iniciales generados aleatoriamente. En cada iteración comienza con un estado inicial aleatorio, lo que permite que se puedan analizar soluciones peores, el algoritmo termina hasta que alcance los criterios de detención.

Como se muestra en el **Algoritmo 9,** el RRHC inicia con una única solución s, tiene un número de iteraciones y un número de reinicios de iteraciones sin mejoras (n_restart). Desde la solución s, un conjunto de nuevas soluciones s' son generadas, y estocásticamente una solución de s' puede ser seleccionada. Luego después de un número determinado de iteraciones sin mejoras, el algoritmo guarda el resultado actual

y se reinicia nuevamente desde otra solución estocástica. El proceso se repite hasta que se cumpla el número de iteraciones.

La adaptación del algoritmo RRHC al problema de etiquetado se realizó de la siguiente manera: 1) la solución se representa como la solución mostrada en la **Figura 5;** 2) la solución se reinicia, seleccionando otra palabra de todas las posibilidades, luego esa palabra se mejora estocásticamente; 3) Se implementó una memoria tabú que guarda las palabras que fueron seleccionadas en el reinicio de la solución, esto con el fin, de no ser seleccionadas en una nueva solución, a menos que anteriormente una palabra vecina haya cambiado su etiqueta. A continuación, en el **Algoritmo 9**, se presenta el pseudocodigo de **RRHCTagger** adaptado al problema de etiquetado.

Algoritmo 9. RRHCTagger

```
1.  Definición del parámetro S, iterations_HC, i_restart y n_restart
2.  i_restart = 0
3.  Prob    /* Lista de las probabilidades de una palabra de acuerdo a todas las opciones */
4.  Inicialización aleatoria de S
5.  ActiveIndex    /* Esta lista guarda las posiciones de las palabras que tienen más de
                      una etiqueta*/
6.  EstadoTrigama          /* Esta lista guarda la posición de las palabras que fueron
                              seleccionadas*/
7.  j = Aleatorio(ActiveIndex.lenght)      /* j va a ser una posición al azar de activeIndex*/
8.  Prob = posiblesProbabilidades(j)        /*obtiene las posibles probabilidades de j*/
9.  X_new ← Copiar(S)
10. para i = 1 to iterations_HC  hacer
11.         si (i_restart > n_restart )              /*Si i_restart supera a n_restart */
12.             EstadoTrigama.add(j)
13.             si (fitness(X_new) > fitness(S))
14.                 S ← X_new
15.                 si (EstadoTrigrama.contains(j − 1)) /*Si el vecino de j se
                                                          encuentra*/
16.                     EstadoTrigrama.remove(j − 1)/*Se elimina de la
                                                      lista */
17.                 fin_si
18.                 si (EstadoTrigrama.contains(j + 1))      /* Si el vecino
                                                              sucesor se encuentra */
19.                     EstadoTrigrama.remove(j + 1)      /* Se elimina de la
                                                           lista */
20.                 fin_si
21.                 X_new ← Copiar(S)        /*Se toma una nueva copia para aplicar la
                                               búsqueda estocástica*/
22.                 pos ← Aleatorio(ActiveIndex.lenght)     /* se selecciona una
                                                             nueva posición de activeIndex
23.                 j = ActiveIndex(pos)   /*La nueva palabra no debe estar en la
                                             lista de EstadoTrigrama */
24.                 Prob = posiblesProbabilidades(j)          /*Se obtienen las
                                                              probabilidades de la nueva palabra*/
25.                 i_restart = 0
26.             fin_si
```

Algoritmo 9. RRHCTagger

```
27.                 Prob.ordenar()          /*Se ordena de forma ascendente, con la mejor
                                              probabilidad en la primera posición */
28.                 X2_new ← Copiar(X_new)  /* Se crea una nueva copia para las
                                              modificaciones que se van a realizar*/
29.                 pos2 =Prob[0]
30.                 X2_new(j) = pos2        /*Se cambia la etiqueta en la posición j*/
31.                 si (fitness(X2_new) > fitness(X_new) >))
32.                       X_new ← X2_new
33.                       Prob.remove(Prob[0]) /* Se elimina esta etiqueta, para no ser
                                                  utilizada de nuevo*/
34.                 else
35.                       Prob.remove(Prob[0])
36.                       i_restart + +
37.                 fin_si
38.   fin_para
39.   retornar S
```

Fuente: Propia

Este algoritmo presenta las siguientes características:

- El parámetro $n\_restart$ controla el número de reinicios de S (línea 11)
- **Prob**, es una lista que almacena las probabilidades de las posibles etiquetas de una palabra (línea 3), esta lista se actualiza cuando se selecciona una nueva palabra a etiquetar (línea 7 y 8), la etiqueta de la palabra j cambia de acuerdo a la etiqueta que tenga mejor probabilidad en **Prob** (línea 29 a la 30), independientemente si $X2_{new}$ fue mejor que X_{new}, la probabilidad que fue usada se elimina de **Prob** (línea 33 y 35).
- **ActiveIndex**, es una lista que almacena las posiciones de las palabras que tienen más de una etiqueta (línea 5).
- **EstadoTrigama**, es una lista que almacena las palabras que ya fueron seleccionadas, para hacer la respectiva mejora estocástica. Si X_{new} mejoro con respecto a S, entonces, si los vecinos se encuentran en **EstadoTrigama,** se remueven de la lista, debido a que el trigrama cambio, y puede haber nuevas probabilidades que sean mejores (línea 15 a la 20).
- Cuando el parámetro $i\_restart$ supera a $n\_restart$, se reinicia la palabra que se quiere etiquetar (línea 11). $i\_restart$ se incrementa cuando $X2_{new}$ no mejora con respecto a X_{new} (línea 36).

Consolidado el algoritmo adaptado **RRHCTagger**, se procede a realizar las pruebas y el ajuste de parámetros para los corpus IULA, Brown y Nasa Yuwe, buscando obtener una mejor versión del algoritmo para cada lengua.

4.2.1 Experimentos realizados para el corpus IULA

Los experimentos sobre el algoritmo **RRHCTagger**, fueron llevados a cabo sobre el data-set del corpus IULA de 5000 frases, presentado en la **Tabla 7**, utilizando validación cruzada de 5 folders.

Para todos los experimentos se definió el parámetro $iterations\_HC$ en 110, que es el número límite de evaluaciones de la función $fitness$. El parámetro $n\_restart$ cambio en cada experimento, buscando el mejor experimento.

Teniendo en cuenta lo escrito anteriormente, se procedió a hacer diferentes experimentos mediante un ajuste de parámetros con la técnica de prueba y error. En la **Tabla 20** se pueden observar los resultados de los experimentos relevantes sobre el corpus IULA.

En la **Tabla 20** se muestra que el experimento 4, con inicialización mejorada y con $n\_restart$ igual a seis, sobrepasa en el valor de precisión a los demás experimentos; por lo tanto, fue el que se ejecutó sobre el data-set completo del corpus IULA, presentado en la **Tabla 4.**

Tabla 20. Resultados de experimentos de 5000 frases para RRHCTagger – Corpus IULA. Fuente: Propia.

#	Algoritmos	n_restart	Frases	Precisión (%)	Desviación estándar
1	RRHCTagger 0.3	5	5000	93.868	7.594
2	RRHCTagger 0.5	4	5000	94.101	7.558
3	RRHCTagger 0.5	5	5000	94.103	7.560
4	**RRHCTagger 0.5**	**6**	**5000**	**94.111**	**7.558**

4.2.2 Experimentos realizados para el corpus Brown

Los experimentos sobre el algoritmo **RRHCTagger**, fueron llevados a cabo sobre el data-set del corpus Brown de 5000 frases, presentado en la **Tabla 7**, utilizando validación cruzada de 5 folders.

Tabla 21. Dataset de 5000 frases corpus Brown. Fuente: Propia.

Carpeta con los datos de prueba	Oraciones en los datos de prueba	Palabras en los datos de prueba	Carpeta con los datos de entrenamiento	Palabras en los datos de entrenamiento	Palabras comunes	Palabras desconocidas
1	1000	4322	2, 3, 4, 5	10662	2789	1533 (35%)
2	1000	4295	1, 3, 4, 5	10707	2807	1488 (35%)
3	1000	4245	1, 2, 4, 5	10687	2737	1508 (36%)
4	1000	4276	1, 2, 3, 5	10674	2755	1521 (36%)
5	1000	4360	1, 2, 3, 4	10616	2781	1579 (36%)

Para la ejecución de **RRHCTagger** sobre el corpus Brown, se procedió a realizar los mismos experimentos presentados en la sección anterior. En la **Tabla 22** se pueden observar los resultados de los experimentos.

Tabla 22. Resultados de experimentos de 5000 frases para RRHCTagger – Corpus Brown. Fuente: Propia.

#	Algoritmos	n_restart	Frases	Precisión (%)	Desviación estándar
1	RRHCTagger 0.3	5	5000	91.296	7.987
2	**RRHCTagger 0.5**	**4**	**5000**	**91.450**	**8.014**
3	RRHCTagger 0.5	5	5000	91.416	8.003
4	RRHCTagger 0.5	6	5000	91.382	7.999

En la **Tabla 22** se muestra que el experimento 2, con inicialización mejorada y con $n\_restart$ igual a cuatro, sobrepasa en el valor de precisión a los demás experimentos; por lo tanto, fue el que se ejecutó sobre el data-set completo del corpus Brown, presentado en la **Tabla 14.**

4.2.3 Experimentos realizados para el corpus Nasa Yuwe

Los experimentos sobre el corpus Nasa Yuwe, para el algoritmo RRHCTagger fueron llevados a cabo utilizando Leave-One-Out, es decir, se utilizó una sola frase como prueba y las otras 174 frases como datos de entrenamiento (175 folders). El conjunto de datos y entrenamientos es el presentado en la **Tabla 11**.

Para la ejecución de RRHCTagger sobre el corpus Nasa Yuwe, se procedió a realizar los mismos experimentos presentados en la sección anterior. En la **Tabla 23**, se pueden observar los resultados de los experimentos.

Tabla 23. Resultados de experimentos de 175 frases para RRHCTagger – Corpus Nasa Yuwe. Fuente: Propia.

#	Algoritmos	n_restart	Frases	Precisión (%)	Desviación estándar
1	RRHCTagger 0.3	5	175	63.453	16.098
2	**RRHCTagger 0.5**	**4**	**175**	**63.709**	**16.224**
3	RRHCTagger 0.5	5	175	63.221	16.195
4	RRHCTagger 0.5	6	175	62.520	16.185

La la **Tabla 23** muestra que el experimento 2, con inicialización aleatoria mejorada y con $n\_restart$ igual a cuatro, la cual mejora el valor de precisión de los demás experimentos, este experimento será comparado con los demás algoritmos del capitulo

4.3 Mejora del algoritmo GBHS2Tagger para el corpus IULA

El contexto utilizado en [25] para el etiquetado con el algoritmo memético GBHS2Tagger, fue un trigrama, es decir, al momento de asignar la etiqueta de la palabra se tenía en cuenta la etiqueta de la palabra predecesora y la de la palabra sucesora, como se muestra en la **Figura 6**.

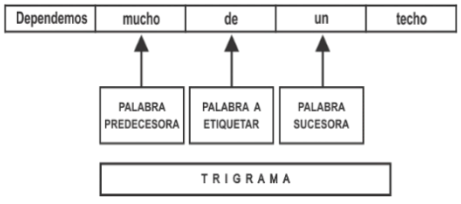


Figura 6. Contexto del Trigrama

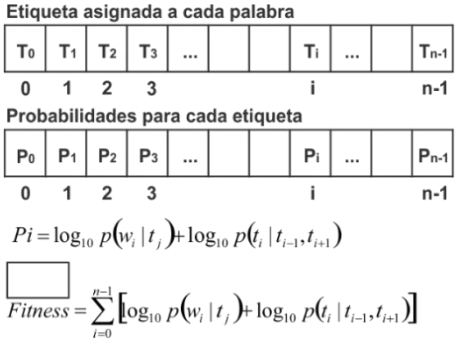


Figura 7. Representación de la solución

La representación de la solución fue organizada como se muestra en la **Figura 7**, así: el primer vector contiene las etiquetas asignadas a cada palabra de la frase a etiquetar, donde T_0, representa la etiqueta de la primera palabra y así sucesivamente para todas las palabras de la frase; el segundo vector contiene cada una de las probabilidades acumuladas para cada etiqueta, representadas en la **Figura 7** como los P_i; finalmente,

se tiene otro campo que almacena el valor de la función fitness, que se calcula como la sumatoria de los logaritmos de los P_i [25].

Teniendo en cuenta esa representación, las mejoras propuestas en la presente investigación, se enfocaron en revisar el desempeño del algoritmo memético GBHS Tagger 2 con diferentes contextos; se seleccionaron 4 contextos que mantienen la representación de la solución de la **Figura 7**, cambiando la forma en que se calcula los P_i del segundo vector y la función fitness, así:

- **Contexto 1,** tiene en cuenta un tetragrama, conformado por dos palabras predecesoras, la palabra a etiquetar y una sucesora; realizando el cálculo de los P_i utilizando la **Ecuación 5**.

$$P_i = log_{10}P(w_i|t_i) + log_{10}P(t_i|\boldsymbol{t_{i-2}}, \boldsymbol{t_{i-1}}, \boldsymbol{t_{i+1}})$$

Ecuación 5. Probabilidad del contexto 1

- **Contexto 2**, también se representa con un tetragrama conformado por: una palabra predecesora, la palabra a etiquetar y dos sucesoras; realizando el cálculo de los P_i utilizando la **Ecuación 6.**

$$P_i = log_{10}P(w_i|t_i) + log_{10}P(t_i|\boldsymbol{t_{i-1}}, \boldsymbol{t_{i+1}}, \boldsymbol{t_{i+2}})$$

Ecuación 6. Probabilidad del contexto 2

- **Contexto 3**, es un trigrama organizado así: la palabra a etiquetar y dos sucesoras; realizando el cálculo de los P_i utilizando la **Ecuación 7**:

$$P_i = log_{10}P(w_i|t_i) + log_{10}P(t_i|\boldsymbol{t_{i+1}}, \boldsymbol{t_{i+2}})$$

Ecuación 7. Probabilidad del contexto 3

- **Contexto 4**, también un trigrama organizado así: dos palabras predecesoras y la palabra a etiquetar; realizando el cálculo de los P_i utilizando la **Ecuación 8**.

$$P_i = log_{10}P(w_i|t_i) + log_{10}P(t_i|\boldsymbol{t_{i-2}}, \boldsymbol{t_{i-1}})$$

Ecuación 8. Probabilidad del contexto 4

El **contexto 5**, hace referencia al trigrama original del algoritmo, es decir, una palabra predecesora, la palabra a etiquetar y una sucesora. El cálculo de los P_i se hace como se representa en la **Figura 7.**

Seguidamente a la definición de los contextos, se configuraron experimentos preliminares que involucraron la construcción de un dataset con 5.000 frases, tomadas de forma aleatoria del dataset descrito en la sección 3.2.1 Construcción de dataset IULA. Los experimentos se ejecutaron utilizando validación cruzada de 5 folders, por ejemplo, si las oraciones del folder 1 fueron tomadas como datos de prueba, las

oraciones de entrenamiento son tomadas desde el folder 2 a 5, y así sucesivamente para todos los folders. Cada folder con 1.000 oraciones de prueba y un promedio de palabras superior a 4.000, de las cuales el 35% o 36% eran desconocidas. Cada contexto se implementó e identificó en el algoritmo GBHS2Tagger así: GBHS2Pred2Suc, para el Contexto 1, GBHS2PredSuc2 Contexto 2, GBHS2Suc2 Contexto 3, GBHS2Pred2 Contexto 4, y finalmente, GBHS2PredSuc Contexto 5. En la **Tabla 24**, se presentan los resultados de los experimentos preliminares.

Tabla 24. Resultados de la ejecución del algoritmo GBHS2 Tagger con diferentes contextos. Fuente: Propia

Contexto	Algoritmo	Número de oraciones	Precisión (%)	Desviación estándar
Contexto 4	**GBHS2Pred2 0.5 0.5**	**5000**	**94.2810022**	**7.897156773**
Contexto 5	GBHS2PredSuce 0.5 0.5	5000	94.14608674	7.702875326
Contexto 2	GBHS2PredSuc2 0.5 0.5	5000	94.09230958	7.68880168
Contexto 1	GBHS2Pred2Suc 0.5 0.5	5000	94.0811572	7.722820905
Contexto 3	GBHS2Suc2 0.5 0.5	5000	94.07903069	7.902208265

Como se puede apreciar, los resultados muestran que el algoritmo GBHS2Pred2 (Contexto 4) supera en precisión a los demás etiquetadores, incluso al trigrama presentado en [25] para el caso del castellano.

Además de estos resultados se realizó una prueba t de student [85], este test se utiliza para determinar si existen diferencias significativas o no entre las medias de dos muestras independientes, a continuación se describe el test realizado. Para nuestro caso se hizo la comparación del contexto 4 con los otros contextos. A continuación, se compara el contexto 4 contra el 5, esto mismo procedimiento se repitió para cada contexto. Las hipótesis a evaluar son:

Ho = las medias son iguales

H1 = las medias son diferentes

En la **Tabla 25**, se muestra las medias de los dos contextos.

Tabla 25. Medias de los contextos. Fuente: Propia

Contexto	Media	Desviación estándar	Varianzas	Muestra
Contexto 4	94.2810	7.8971	62.3650	n1 =30
Contexto 5	94.1460	7.7028	59.3342	n2 =30

Para realizar el test se aplica la expresión de la t de student, presentada en la **Ecuación 9** para comparar las dos medias, en dónde, $EEDM$ es el "error estándar de la diferencia de medias".

$$t_{n+m-2} = \frac{\overline{x_1} - \overline{x_2}}{EEDM}$$

Ecuación 9. Expresión de la t de Student. Fuente: [85].

Para calcular él $EEDM$ se hace como se muestra en la **Ecuación 10**, donde S_p es la desviación estándar ponderada, que se calcula como se muestra en la **Ecuación 11**, donde n y m son los individuos para cada grupo respectivamente.

$$EEDM = S_p \sqrt{\frac{1}{n} + \frac{1}{m}}$$

Ecuación 10. Error estándar de la diferencia de medias**. Fuente:** [85].

$${S_p}^2 = \frac{(n-1){S_1}^2 + (m-1){S_2}^2}{(n-1) + (m-1)}$$

Ecuación 11. Desviación estándar ponderada**. Fuente:** [85].

Teniendo todos los datos, se calculó cada una de las ecuaciones con los siguientes resultados $\boldsymbol{S_p = 7.800620917}$, $\boldsymbol{EEDM = 2.01411166}$ y $\boldsymbol{t_{58} = 0.066985095}$, para nuestro caso t tiene 58 grados de libertad.

Este valor de t no es significativo ya que el valor tabulado para un error $\boldsymbol{\propto = 0.05}$ es superior al encontrado, el valor de t tabulado se tomó de la tabla de la distribución t [85], que es igual a $\boldsymbol{t_{58,\frac{\propto}{2}} = 0.025 = 2.000}$, entonces como $\boldsymbol{t_{58} = 2.001717484 > 0.066985095}$, el resultado del test muestra que no se encuentran diferencias estadísticas entre las medias obtenidas, es decir, Ho es verdadera.

Con este resultado orientador (no definitivo), el paso seguido fue la ejecución del algoritmo en su versión mejorada GBHS2Pred2 con el data set completo del corpus IULA, utilizando igualmente validación cruzada de 5 folders, los resultados se muestran en la **Tabla 29** de la sección 4.5 Experimentos realizados sobre los mejores Algoritmos metaheurísticos, donde se pueden apreciar las diferencias con el algoritmo memético GBHS 2 Tagger propuesto en [25].

4.4 Propuesta de un algoritmo memético para el problema de etiquetado

Como se observó en el capítulo 3, el algoritmo metaheurístico GBHS Tagger propuesto en [25], superó a las demás metaheurísticas y fue seleccionado para proponer una nueva versión memética, basada en el algoritmo GBHS como optimizador global y HC

como optimizador local. La versión de GBHS que fue utilizada es la propuesta por [25], el cual ya se encuentra adaptado al problema de etiquetado y la versión de HC presentada en la sección 4.1 Adaptación del algoritmo metaheurístico Hill Climbing al problema de etiquetado.

En el algoritmo **Algoritmo 10** se presenta el algoritmo GBHS adaptado al problema de etiquetado, la armonía se presenta como se muestra en la **Figura 5**, el optimizador local HC se adaptó a GBHS.

Algoritmo 10. GBHS adaptado al problema de etiquetado

1. Definir parámetros: HMS, NI, HCMR, PARMin, ParMax, ProbOpt, Alpha, MaxNeighbors, Prob
2. Inicialización aleatoria de HM o inicialización mejorada usando el parámetro Alpha
3. **para** $i = 1$ to NI **hacer**
4. $PAR \leftarrow PARMin + (PARMax - PARMin) \times \left({}^{i}/_{NI}\right)$ /* definición de PAR */
5. **para** $j = 1$ to n **hacer** /* para cada palabra en la oración */
6. **si** (Activo[j] == true) **entonces** /* ¿la palabra actual tiene más de una posible etiqueta? */
7. **si** $(U(0,1) \leq HMCR)$ **entonces** /* memory consideration */
8. $x_j' \leftarrow x_j^p$, donde $p \sim U(1, \dots, HMS)$
9. **Si** $(U(0,1) \leq PAR$ **entonces**
10. $x_j' \leftarrow x_k^{best}$, donde best es el índice de la mejor armonía en HM and $k \sim U(1, n)$
fin_si
11. **sino** /* selección aleatoria */
12. $x_j' \leftarrow LB_j + r \times (UB_j - LB_j)$
13. **fin_si**
14. **sino**
15. $x_j' \leftarrow$ UnicaEtiquetaParaLaPalabra(j)
16. **fin_si**
17. **fin_para**
18. **mientras** (visitado (x_j')) **hacer** /* Si la solución ha sido visitada antes */
19. **si** (Activo[j] == true) **entonces /*** mutar la nueva armonía (x_j')
20. Se selecciona aleatoriamente una diferente a la actual
21. **fin_mientras**
22. Evaluar la función fitness de la nueva armonía (x_j')
23. **si** (fitness (x_j') > fitness de la peor armonía en HS) **entonces**
24. HM[pos_peor] $\leftarrow x_j'$ /* reemplazar la peor en la memoria armónica */
25. **fin_si**
26. **si** $(U(0,1) < ProbOpt)$ **entonces**
27. Aplicar Optimización Local a la mejor armonía en HM
28. **fin_si**
29. **fin_para**
30. **retornar** la mejor armonía en HM

Este algoritmo presenta las siguientes características:

- Usa los mismos parámetros de su versión original (GBHS), como son: HMCR, PARMin, PARMax, HMS, y NI. El valor de estos parámetros son los mismos que se definieron en [25].

- El número de vecinos (MaxNeighbors) es el número de veces que se evaluara el proceso de optimizador local, para el caso de HC, MaxNeighbors es el parámetro N.
- El parámetro de probabilidad de optimización (ProbOpt), controla el porcentaje de veces que se realiza el proceso de optimización local, este proceso se aplica a la mejor armonía (solución) en la memoria armónica.
- El parámetro Alpha, el cual controla si los componentes de cada armonía en la población son generados aleatoriamente de sus posibles etiquetas o tomados de la etiqueta con mayor probabilidad.
- En la línea 27 se llama al algoritmo HC adaptado al algoritmo GBHSTagger, el tiene como solución inicial la mejor armonía y a partir de ahí se aplica el HC presentado en el **Algoritmo 8.**

Consolidado este algoritmo memético el cual se nombró **GBHS4Tagger**, se procedió a realizar experimentos preliminares sobre los corpus IULA, Brown y Nasa Yuwe.

4.4.1 Experimentos realizados para el corpus IULA

Los experimentos sobre el algoritmo **GBHS4Tagger**, fueron llevados a cabo sobre el data-set del corpus IULA de 5000 frases, presentado en la **Tabla 7**, utilizando validación cruzada de 5 folders.

La configuración de los experimentos consistió en variar el parámetro $Prob$, los parámetros de inicialización mejorada (Alpha) y optimización (ProbOpt) para los experimentos fueron de 0.5, debido a que estos valores fueron los mejores en [25].

Teniendo en cuenta lo escrito anteriormente, se procedió hacer diferentes experimentos mediante un ajuste de parámetros con la técnica de prueba y error. En la **Tabla 13** se pueden observar los resultados de los experimentos relevantes sobre el corpus IULA.

Tabla 26. Resultados de experimentos de 5000 frases paraGBHS4Tagger – Corpus IULA. Fuente: Propia

#	Algoritmos	Prob	Frases	Precisión (%)	Desviación estándar
1	GBHS4Tagger 0.5 0.5	0.5	5000	94.267	7.648

#	Algoritmos	Prob	Frases	Precisión (%)	Desviación estándar
2	**GBHS4Tagger 0.5 0.5**	**0.7**	**5000**	**94.314**	**7.635**
3	GBHS4Tagger 0.5 0.5	0.75	5000	94.309	7.625

En la **Tabla 26**, se muestra que el experimento 2, con inicialización mejorada y el parámetro $Prob = 0.7$, sobrepasa en el valor de precisión a los demás experimentos; por lo tanto, fue el que se ejecutó sobre el data-set completo del corpus IULA, presentado en la **Tabla 4**.

4.4.2 Experimentos realizados para el corpus Brown

Los experimentos sobre el algoritmo **GBHS4Tagger**, fueron llevados a cabo sobre el data-set del corpus Brown de 5000 frases, presentado en la **Tabla 7**, utilizando validación cruzada de 5 folders.

Para la ejecución de **GBHS4Tagger** sobre el corpus Brown, se procedió a realizar los mismos experimentos presentados en la sección anterior. En la **Tabla 27**, se pueden observar los resultados de los experimentos.

Tabla 27. Resultados de experimentos de 5000 frases paraGBHS4Tagger – Corpus Brown. Fuente: Propia

#	Algoritmos	Prob	Frases	Precisión (%)	Desviación estándar
1	GBHS4Tagger 0.5 0.5	0.5	5000	91.246	7.977
2	**GBHS4Tagger 0.5 0.5**	**0.7**	**5000**	**91.283**	**7.958**
3	GBHS4Tagger 0.5 0.5	0.75	5000	91.254	7.960

En la **Tabla 27**, se muestra que el experimento 2, con inicialización mejorada y el parámetro $Prob = 0.7$, sobrepasa en el valor de precisión a los demás experimentos; por lo tanto, este experimento será comparado con los demás algoritmos en la sección 4.5 Experimentos realizados sobre los mejores Algoritmos metaheurísticos

4.4.3 Experimentos realizados para el corpus Nasa Yuwe

Los experimentos sobre el corpus Nasa Yuwe, para el algoritmo **GBHS4Tagger** fueron llevados a cabo utilizando Leave-One-Out, es decir, se utilizó una sola frase como prueba y las otras 174 frases como datos de entrenamiento (175 folders). El conjunto de datos y entrenamientos es el presentado en la **Tabla 11**.

Para la ejecución de **GBHS4Tagger** sobre el corpus Nasa Yuwe, se procedió a realizar los mismos experimentos presentados en la sección anterior. En la **Tabla 28**, se pueden observar los resultados de los experimentos.

Tabla 28. Resultados de experimentos de 175 frases paraGBHS4Tagger – Corpus Nasa Yuwe. Fuente: Propia

#	Algoritmos	Prob	Frases	Precisión (%)	Desviación estándar
1	GBHS4Tagger 0.5 0.5	0.5	5000	60.922	15.401
2	GBHS4Tagger 0.5 0.5	0.7	5000	61.288	15.707
3	**GBHS4Tagger 0.5 0.5**	**0.75**	**5000**	**61.418**	**15.519**

En la **Tabla 28**, se muestra que el experimento 3, con inicialización mejorada y el parámetro $Prob = 0.75$, sobrepasa en el valor de precisión a los demás experimentos; por lo tanto, fue el que se ejecutó sobre el data-set completo del corpus Brown, presentado en la **Tabla 9**.

4.5 Experimentos realizados sobre los mejores Algoritmos metaheurísticos

En esta sección se presenta los resultados de la ejecución de los etiquetadores GBHS4Tagger y RRHCTagger sobre los datasets completos de los corpus IULA, Brown y Nasa Yuwe, en pro de seleccionar el mejor algoritmo, basado en la medida de precisión y comparados con el algoritmo GBHS2Tagger. Además, se presenta la ejecución de la mejora GBHS2Pred2 sobre el dataset completo del corpus IULA.

4.5.1 Resultados del mejor algoritmo metaheurístico para el corpus IULA

Los experimentos para el corpus IULA se ejecutaron con los algoritmos metaheurísticos GBHS4Tagger, RRHCTagger y GBHS2Pred2, utilizando el data-set completo, presentado en la **Tabla 4**; cada experimento fue configurado de la siguiente manera:

- **GBHS4Tagger:** se tomó el experimento número 2, presentado en la **Tabla 26** con el parámetro $Prob = 0.7$, que fue el que superó en precisión a los demás experimentos, para su ejecución se utilizó validación cruzada de 5 folders.

- **RRHCTagger:** se tomó el experimento número 4 de la **Tabla 20** con inicialización mejorada y $n\_restart$ igual a seis, el cual superó en precisión a los demás experimentos, para su ejecución se utilizó validación cruzada de 5 folders.

- **GBHS2Pred2:** se tomó el experimento que utilizo el contexto 4 de la **Tabla 24,** que utilizó validación cruzada de 5 folders.

En la **Tabla 29**, se presentan los resultados de los experimentos realizados para cada algoritmo, donde se incluyó el resultado del etiquetador GBHS Tagger2 presentado en la línea base, con el fin de hacer la comparación de los mejores algoritmos.

Tabla 29. Resultados de experimentos para el Corpus IULA – Mejores algoritmos. Fuente: Propia.

Algoritmo	Frases	Precisión	Desviación estándar	Precisión (%) palabras desconocidas	Desviación estándar
GBHS4Tagger	**42079**	**97.5403**	**5.4795**	**94.1986**	**7.4530**
RRHCTagger	42079	97.2678	5.5006	94.0865	7.4208
GBHS2Pred2	**42079**	**97.6051**	**5.6494**	**94.4238**	**7.8071**
GBHS Tagger2	42079	97,4306	5,7844	94.1928	7.8055

En la **Tabla 29**, se puede observar que el algoritmo GBHS2Pred2 superó a los demás algoritmos en valor de precisión. Adicionalmente, en los resultados se destaca el algoritmo RRHCTagger con un valor cercano a los demás algoritmos de 94.1986, este resultado es muy bueno teniendo en cuenta que es una metaheurística con una única solución.

Además de los resultados presentados en la **Tabla 29**, se aplicaron dos tests estadísticos, que nos permitieron ver las diferencias entre los algoritmos. En las dos siguientes secciones se muestran los resultados de cada test.

1. Test de Friedman

En la **Tabla 30,** se muestran los puntajes obtenidos para cada algoritmo, según el test de Friedman NXN, el algoritmo de etiquetado GBHS2Pred2 se posiciona como el mejor para el corpus IULA. Este test tiene 3 grados de libertad (15) y un valor de p (p-value) de 0.0018, que lo hace estadísticamente significativo.

Tabla 30. Resultados Test de Friedman – Corpus IULA. Fuente: Propia de software KEEL [79]

Algoritmo	Ranking Friedman
GBHS2Pred2 0.5 0.5	1
GBHS4Tagger 0.5 0.5 0.7	2
GBHS Tagger2 0.5 0.5	3
RRHCTagger 0.5 6	4

2. Test de Wilconson

La aplicación del test mostró con un 90% de significancia que los resultados obtenidos con el algoritmo GBHS2Pred2 son mejores que los otros algoritmos presentados en la Tabla 29, adicionalmente se pudo observar que el algoritmo GBHS4Tagger supera a el algoritmo GBHS Tagger 2.

4.5.2 Resultados del mejor algoritmo metaheurístico para el corpus Brown

Los experimentos para el corpus Brown se ejecutaron con los algoritmos metaheurísticos GBHS4Tagger, RRHCTagger y GBHS Tagger 2, utilizando el data-set completo, presentado en la **Tabla 4**; cada experimento fue configurado de la siguiente manera:

- **GBHS4Tagger:** se tomó el experimento número 2, presentado en la **Tabla 27** con el parámetro $Prob = 0.7$, que fue el que superó en precisión a los demás experimentos, para su ejecución se utilizó validación cruzada de 5 folders.

- **RRHCTagger:** se tomó el experimento número 2 de la **Tabla 22**, con inicialización mejorada y $n\_restart$ igual a cuatro, el cual superó en precisión a los demás experimentos, para su ejecución se utilizó validación cruzada de 5 folders.

- **GBHS Tagger 2:** se replicó el experimento hecho en [25], bajo las mismas condiciones de los otros experimentos, es decir, la misma semilla, con los mismos ordenadores y utilizando validación cruzada de 5 folders.

En la **Tabla 31**, se presentan los resultados de los experimentos realizados para cada algoritmo, donde se replicó el experimento del etiquetador GBHS Tagger2 presentado en la línea base, con el fin de hacer la comparación de los mejores algoritmos en las mismas condiciones.

Tabla 31. Resultados de experimentos para el Corpus Brown – Mejores algoritmos. Fuente: Propia.

Algoritmo	Frases	Precisión	Desviación estándar	Precisión (%) palabras desconocidas	Desviación estándar
GBHS4Tagger	**52998**	**94.8803**	**6.0289**	**92.4432**	**6.3016**
RRHCTagger	52998	94.5342	6.0106	92.1588	6.3228
GBHS Tagger2	52998	94.8236	6.2051	92.3196	6.3888

En la **Tabla 31**, se puede observar que el algoritmo GBHS4Tagger superó a los demás algoritmos en valor de precisión. No obstante, en los resultados se destaca el algoritmo

RRHCTagger con un valor cercano a los demás algoritmos de 94.5324, este resultado es muy bueno teniendo en cuenta que es una metaheurística con una única solución.

Además de los resultados presentados en la **Tabla 31**, se aplicaron dos tests estadísticos, que nos permitieron ver las diferencias entre los algoritmos. En las dos siguientes secciones se muestran los resultados de cada test.

1. Test de Friedman

En la **Tabla 32,** se muestran los puntajes obtenidos para cada algoritmo, según el test de Friedman NXN, el algoritmo de etiquetado GBHS4Tagger se posiciona como el mejor para el corpus Brown. Este test tiene 2 grados de libertad (10) y un valor de p (p-value) de 0.0067, que lo hace estadísticamente significativo.

Tabla 32. Resultados Test de Friedman – Corpus Brown. Fuente: Propia de software KEEL [79]

Algoritmo	Ranking Friedman
GBHS4Tagger 0.5 0.5 0.7	1
GBHS Tagger 2 0.5 0.5	2
RRHC Tagger 0.5 6	3

2. Test de Wilconson: la aplicación del test mostro con un 90% de significancia que los resultados obtenidos con el algoritmo GBHS4Tagger son mejores que los otros algoritmos presentados en la Tabla 31**.**

4.5.3 Resultados del mejor algoritmo metaheurístico para el corpus Nasa Yuwe

Para la comparación del corpus Nasa Yuwe, se seleccionó el mejor experimento para cada algoritmo metaheurístico, así:

- **GBHS4Tagger:** se tomó el resultado del experimento número 3, presentado en la **Tabla 28** con el parámetro $Prob = 0.75$, que fue el que superó en precisión a los demás experimentos, para su ejecución se utilizó validación cruzada de 5 folders.

- **RRHCTagger:** se tomó el resultado del experimento número 2 de la **Tabla 22**, con inicialización mejorada y $n\_restart$ igual a cuatro, el cual superó en precisión a los demás experimentos, para su ejecución se utilizó validación cruzada de 5 folders.

- **GBHS Tagger2:** se replicó el experimento hecho en [26], bajo las mismas condiciones, es decir, la misma semilla, con los mismos ordenadores y utilizando la técnica Leave-One-Out.

En la **Tabla 33**, se presentan los resultados de los experimentos realizados para cada algoritmo, donde se replicó el experimento del etiquetador GBHS Tagger2 presentado en la línea base, con el fin de hacer la comparación de los mejores algoritmos.

Tabla 33. Resultados de experimentos para el Corpus NasaYuwe – Mejores algoritmos. Fuente: Propia.

Algoritmo	Frases	Precisión	Desviación estándar	Precisión (%) palabras desconocidas	Desviación estándar
GBHS4Tagger	175	61.4185	15.5199	62.4022	14.7173
RRHCTagger	**175**	**63.7096**	**16.2249**	**63.6688**	**15.5570**
GBHS Tagger2	175	60.0749	15.5199	61.2142	15.9936

En la **Tabla 33**, se puede observar que el algoritmo RRHCTagger superó a los demás algoritmos en valor de precisión, este algoritmo de estado simple utiliza una única solución.

Además de los resultados presentados en la **Tabla 33**, se aplicaron dos tests estadísticos, que nos permitieron ver las diferencias entre los algoritmos. En las dos siguientes secciones se muestran los resultados de cada test.

1. Test de Friedman

En la **Tabla 34,** se muestran los puntajes obtenidos para cada algoritmo, según el test de Friedman NXN, el algoritmo de etiquetado RRHCTagger se posiciona como el mejor para el corpus Brown. Este test tiene 2 grados de libertad (24.4914) y un valor de p (p-value) de 4.805692676468354E-6, que lo hace estadísticamente significativo.

Tabla 34. Resultados Test de Friedman – Corpus Nasa Yuwe. Fuente: Propia de software KEEL [79]

Algoritmo	Ranking Friedman
RRHCTagger 0.5 4	1
GBHS4Tagger 2 0.5 0.5 0.7	2
GBHS Tagger2 0.5 0.5	3

2. Test de Wilconson:

La aplicación del test mostro con un 90% de significancia que los resultados obtenidos con el algoritmo RRHCTagger son mejores que los otros algoritmos presentados en la **Tabla 34,** adicionalmente se pudo observar que el algoritmo GBHS4Tagger supera a el algoritmo GBHS Tagger 2.

4.7 Síntesis

En el presente capítulo, se pudo apreciar que las metaheurísticas de estado simple como HC y RRHC obtienen muy buenos resultados, incluso sobrepasando a

metaheurísticas poblaciones, como se pudo observar en los experimentos realizados de RRHC sobre el corpus Nasa Yuwe, además son más simples de implementar y más rápidas en la ejecución de experimentos, ya que utilizan una única solución.

Otro aspecto importante es que, en la búsqueda de mejores resultados de precisión, modificar un algoritmo, depende mucho de la lengua a etiquetar. En los algoritmos se pueden aplicar muchas formas de optimización que aprovechan una determinada lengua, pero para otras lenguas pueden ser una desventaja, esto explica porque en algunos experimentos puede variar los parámetros dependiendo de cada lengua, debido a esto es importante hacer experimentos preliminares con datasets pequeños de los corpus y así definir un mejor experimento, como se hizo en esta investigación.

Un resultado importante de este capítulo fue un artículo titulado "*Etiquetado de partes del discurso sobre un corpus en castellano basado en metaheurísticas*", el cual fue aceptado para presentación en el III COISINT 2020 y su publicación en la Revista RISTI. Este articulo presenta la construcción del dataset IULA para el etiquetado y la línea base para el castellano presentado en la sección **3.2.2** Experimentos con dataset IULA del capítulo 3, además se incluyó la mejora al algoritmo GBHS Tagger2 descrita en la sección **4.3** Mejora del algoritmo GBHS2Tagger para el corpus IULA del presente capitulo. En el anexo A, se encuentra el articulo enviado, junto con la carta de aceptación. El paper se encuentra pendiente de publicación.

En el anexo digital 2 (shorturl.at/mBDHU), se encuentran los algoritmos adaptados e implementados en este trabajo.

Capítulo 5

Desarrollo de un servicio web y una aplicación web para la prueba de los algoritmos presentados.

En este capítulo se presenta un servicio web y una aplicación web, utilizando metodología ágil SCRUM, con su respetiva documentación; para el desarrollo de estos dos artefactos se estableció una pila del producto (Product Backlog), que desencadenó la definición de 2 Sprint, cada uno con sus respectivas historias de usuario y criterios de validación.

5.1 Pila del producto (Product Backlog) y Sprint

5.1.1 Definición de la Pila del producto – Product Backlog

La pila del producto se estableció con todos los requerimientos necesarios para la construcción del servicio web y la aplicación web, contemplando la arquitectura y la base de datos. En la **Tabla 35** se presentan las historias épicas y su respectiva descripción y prioridad.

Tabla 35. Pila del producto – Etiquetado de partes del discurso. Fuente: Propia

Id	Requerimiento	Prioridad
R1	Desarrollar una Api Web que permita a un cliente http realizar el etiquetado POST y devuelva su resultado	1
R2	Desarrollar una aplicación web adaptativa que permita a un usuario seleccionar una frase de los corpus IULA, Brown y Nasa Yuwe para posteriormente hacer una petición al api de ejecutar el etiquetado.	2

Teniendo en cuenta la pila del producto, presentada en la **Tabla 35**, para la historia épica R1, se definieron las historias de usuario mostradas en la **Tabla 36,** que se ordenaron según su prioridad**.**

Tabla 36. Historias de Usuario – R1. Fuente: Propia

Id	Prioridad	Descripción	Criterios de Aceptación
HU01	1	**Como:** Desarrollador **Quiero:** Hacer peticiones a una Api **Para:** comunicar el frontend con el backend	**Criterio 1** **Dado:** Que se requiera hacer una petición **Cuando:** Una aplicación hace una petición al api **Entonces:** El api retorna una respuesta
HU02	2	**Como:** Aplicación web **Quiero:** Consultar las frases que coinciden con un texto determinado	**Criterio 1** **Dado:** Que tenga un texto en Ingles, Castellano o Nasa Yuwe

Id	Prioridad	Descripción	Criterios de Aceptación
		Para: listar las frases que coinciden	**Cuando:** una aplicación envié el texto al api **Entonces:** El api devuelve las frases que coinciden
			Criterio 1 **Dado:** Que tenga un texto en Ingles, Castellano o Nasa Yuwe **Cuando:** una aplicación envié el texto al api **Entonces:** El api devuelve las frases que coinciden
HU03	3	**Como:** Aplicación web **Quiero:** Ejecutar un algoritmo metaheurístico **Para:** mostrar el resultado del etiquetado	**Criterio 1** **Dado:** Que requiera ejecutar un algoritmo **Cuando:** una aplicación envié una petición get **Entonces:** El api devuelve las frases que coinciden
HU04	4	**Como:** Servicio web **Quiero:** Conectarme a la base de datos de los corpus **Para:** Gestionar las peticiones a la base de datos	**Criterio 1** **Dado:** Que requiera hacer una consulta a la base de datos **Cuando:** el servicio web necesite consultar o afectar la base de datos **Entonces:** la base de datos retorna el resultado de la consulta

.

Para la historia épica R2, se definieron las historias de usuario mostradas en la **Tabla 37.**

Tabla 37. Historias de Usuario – R2. Fuente: Propia.

Id	Prioridad	Descripción	Criterios de Aceptación
HU05	1	**Como:** Usuario **Quiero:** Seleccionar una frase de la lengua Castellano, Inglés o Nasa Yuwe **Para:** posteriormente ejecutar el proceso de etiquetado	**Criterio 1** **Dado:** Que tenga un texto en Castellano, Inglés o Nasa Yuwe **Cuando:** El usuario digite el texto **Entonces:** La aplicación muestra las coincidencias del texto.
			Criterio 1 **Dado:** Que se muestra una frase en la lista de coincidencias **Cuando:** El usuario selecciona la frase **Entonces:** La aplicación muestra la frase en un campo de texto
HU06	2	**Como:** Usuario **Quiero:** Seleccionar un algoritmo de la lista de posibilidades **Para:** posteriormente ejecutar el proceso de etiquetado para ese algoritmo	**Criterio 1** **Dado:** Que se seleccione una frase **Cuando:** El usuario selecciona un algoritmo para ejecutar **Entonces:** Se habilita el botón para ejecutar el proceso de etiquetado
HU07	3	**Como:** Usuario	**Criterio 1**

Id	Prioridad	Descripción	Criterios de Aceptación
		Quiero: realizar el etiquetado de una frase de la lengua castellano, Inglés o Nasa Yuwe **Para:** Saber las etiquetas léxicas de la frase	**Dado:** Que se seleccione una frase para etiquetar y se elija un algoritmo **Cuando:** Se presiona el botón de etiquetar **Entonces:** La aplicación muestra que el proceso de etiquetado está en ejecución
			Criterio 2 **Dado:** Que se esté corriendo el proceso de etiquetado **Cuando:** Termine el proceso **Entonces:** La aplicación muestra el resultado del etiquetado y las etiquetas para cada palabra

La ejecución de las historias de usuario que describen la pila del producto se planeó en dos sprints, descritos a continuación.

5.1.2 Planeación del Sprint 1 – Servicio Web

Para el sprint 1 se desarrollaron en orden de prioridad las historias de usuario presentadas en la **Tabla 36,** estas comprenden el desarrollo del servicio web, en el cual estan los algoritmos metaheurísticos que ejecutan el etiquetado POST, sobre los corpus IULA, Brown y Nasa Yuwe.

5.1.3 Planeación del Sprint 2 – Aplicación Web

Para el sprint 2 se desarrollaron en orden de prioridad las historias de usuario presentadas en la **Tabla 37,** estas contemplan el desarrollo de la aplicación web que consume el servicio web, permitiendo a un usuario etiquetar frases de los corpus IULA, Brown y Nasa Yuwe y se mostrará los resultados.

5.2 Desarrollo del Servicio Web – Sprint 1

Definidas las historias de usuario, se inició el sprint 1, donde la primera actividad fue la definición de la arquitectura del servicio web, para la cual se utilizó una arquitectura API REST [86], la duración de este sprint fue de 1 semana. En la **Figura 8**, se presenta los componentes que interactúan con el API.

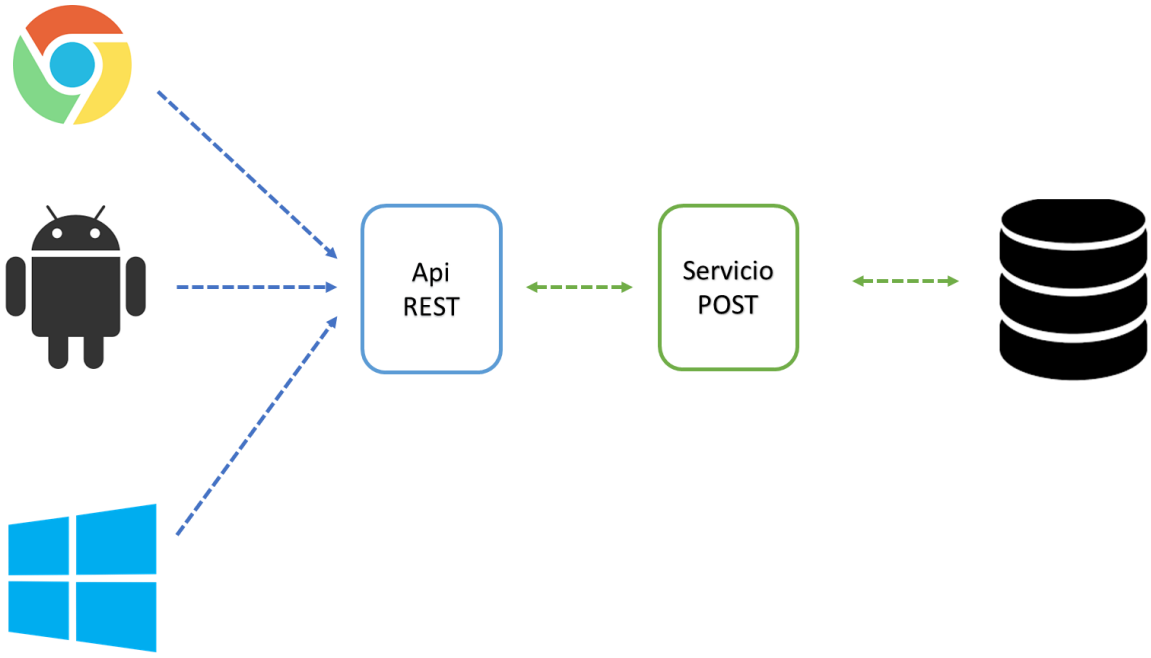


Figura 8. Arquitectura de la Api. Fuente: Adaptado de [86]

Luego de definir la arquitectura, se realizó la codificación de cada historia de usuario, el desarrollo del api se hizo en el framework ASP.NET y en el lenguaje de programación C#, donde cada día se cumplió con el Scrum diario para hacer el seguimiento del sprint 1. Luego del desarrollo del API, se realizaron las pruebas de unidad e integración de los requisitos implementados en el sprint 1.

Para las pruebas de la Api se utilizó la aplicación Insomnia [87], el cual es una cliente que permite realizar peticiones REST permitiendo verificar el comportamiento de una Api, para este caso se realizaron casos de prueba presentados en la **Tabla 38**.

Tabla 38. Casos de prueba Sprint 1. Fuente: Propia.

ID Caso prueba	Título del caso de prueba	Precondiciones	Acción	Resultado esperado
TC_01	Coincidencias de frase	1. El api debe estar en ejecución 2. Abrir Insomnia	1. Seleccionar el tipo request Get 2. Digitar la url base del servicio. Ej: 127.0.0.1:3000 3. Concatenar a la url la palabra "Frase?cadena=" 4. Concatenar a la url el texto a buscar, ej: prueba 5. clic en "Send"	El api retorna un json con una lista de frases que coinciden con el texto descrito en el paso 4

ID Caso prueba	Título del caso de prueba	Precondiciones	Acción	Resultado esperado
TC_02	Resultado de ejecución de algoritmo	1. El api debe estar en ejecución 2. Abrir Insomnia	1. Seleccionar el tipo request Get 2. Digitar la url base del servicio. Ej: 127.0.0.1:3000 3. Concatenar a la url la palabra "Frase?corpus=" 4. Concatenar a la url el numero el id del corpus ej: 1 5. Concatenar a la url la palabra "&Algoritmo=" 6. Concatenar a la url el numero el id del corpus ej: PSOTagger 7. Concatenar a la url la palabra "&fraseid=" 8. Concatenar a la url el numero el id de la frese ej: 904 5. clic en "Send"	El api retorna un json con los campos correspondientes al resultado de la frase del paso 8, ejecutada bajo el algoritmo del paso 5

Para cumplir con los criterios de aceptación de cada historia de usuario, a continuación, se describe las pruebas realizadas:

- Para HU1 se utilizó el request [getPrueba] de prueba para verificar el correcto funcionamiento del Api, en la **Figura 9** se muestra el resultado del api.

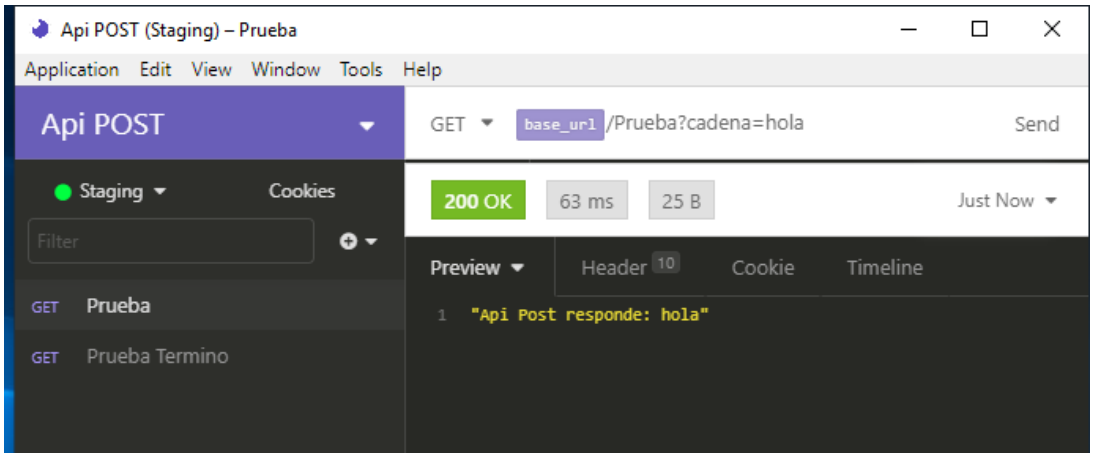


Figura 9**.** Request 1. Fuente: Propia

- Para HU4 se utilizó el request [Prueba], pero esta vez realizando una consulta a la base de datos desde la api, en la **Figura 10** se muestra el resultado del API.

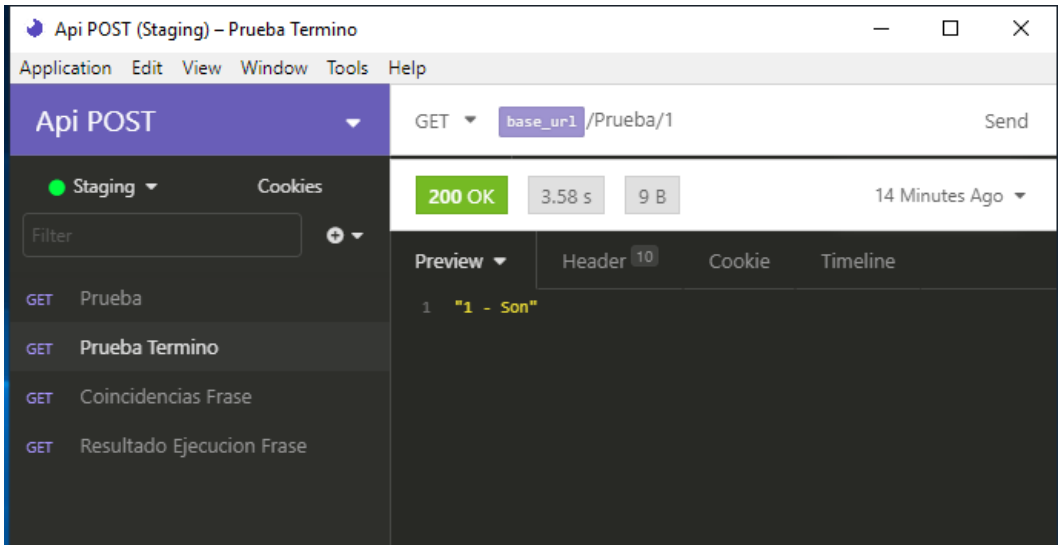


Figura 10. Request 2. Fuente: Propia

- Para HU2 se utilizó el request [Coincidencias Frase] y se ejecutó el caso de prueba TC_01 mostrado en la **Tabla 38**, que retorna una lista con las frases que coinciden con el texto de entrada y sus identificadores, en la **Figura 11** se muestra el resultado del api.

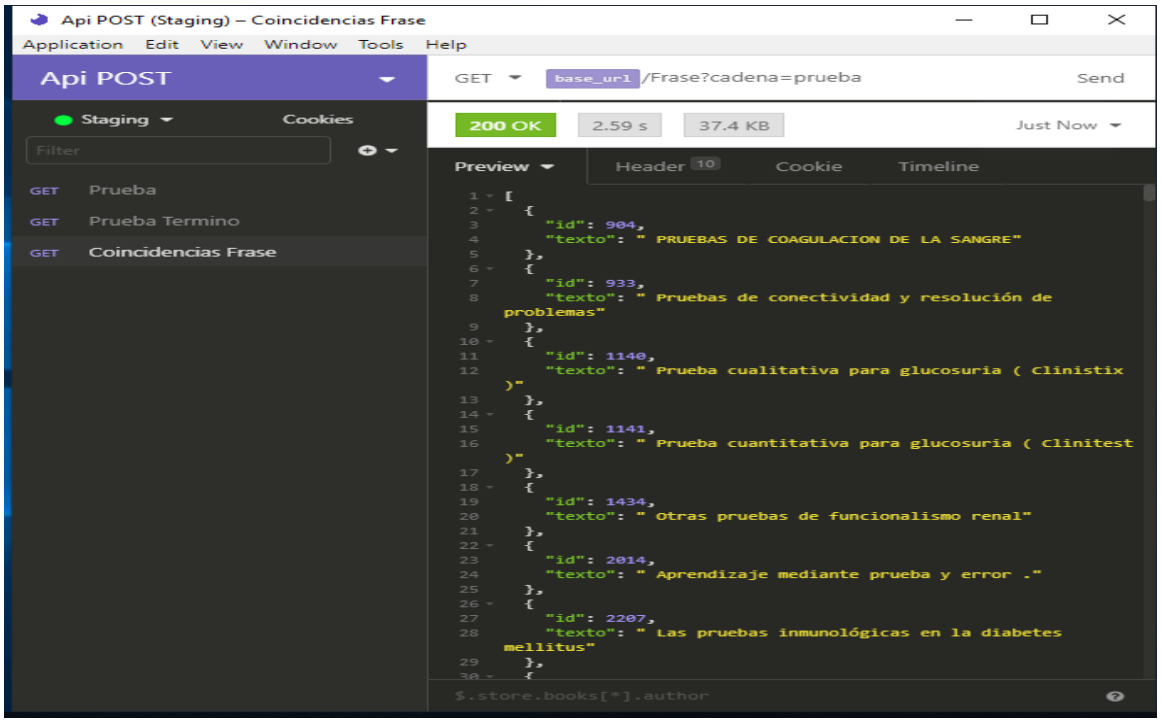


Figura 11**.** Request 3. Fuente: Propia

- Para HU3 se utilizó el request [Resultado Ejecucion frase] y se ejecutó el caso de prueba TC_02 mostrado en la **Tabla 38**, que retorna el resultado de la ejecución de algoritmo determinado, en la **Figura 12** se muestra el resultado del API.

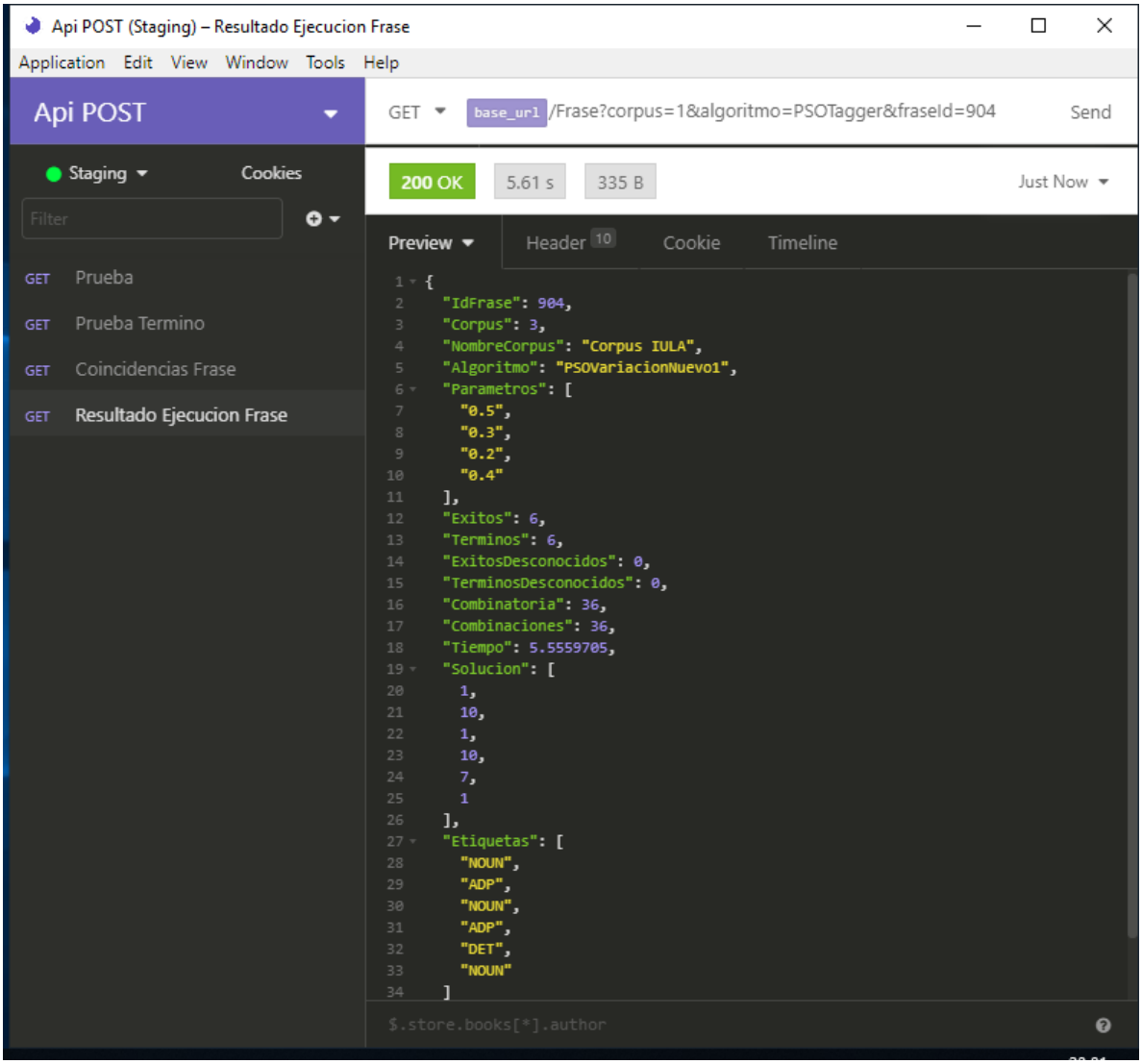


Figura 12. Request 4. Fuente: Propia

Realizadas las respectivas pruebas, se revisó el estado del sprint y el incremento, donde se cumplió con todo lo planificado. Por último, se realizó la retrospectiva del sprint 1, que fue presentada en un formulario, con el objetivo de documentar y que sirva como mejora continua para el siguiente sprint, en la **Tabla 39** se presenta el formulario.

Tabla 39. Reunión de retrospectiva – sprint 1. Fuente: adaptado de [34]

¿Que salió bien en la iteración (aciertos)?	¿Qué no salió bien en la iteración?	¿Qué mejoras vamos a implementar en la próxima iteración? (recomendaciones de mejora continua)
Se cumplió con lo estimado	El tiempo en las pruebas duró más de lo estimado	Mejor descomposición de tareas Mejorar en la comunicación del equipo

5.3 Desarrollo de la Aplicación Web – Sprint 2

Luego de cumplir el sprint 1, se inició el sprint 2, donde la primera actividad fue la definición de la arquitectura de la aplicación web. Como se pudo observar en la **Figura 8**, los clientes se comunicarán con la API para realizar las respectivas peticiones, para nuestro caso se decidió desarrollar una aplicación web.

La aplicación web se desarrolló bajo el lenguaje Javascript utilizando la librería React js [88], diseñada para crear interfaces de usuario, la cual es basada en componentes que manejan su propio estado y así reutilizar en cualquier interfaz que se requiera, en la **Figura 13** se presenta la arquitectura de componentes de react y como se comunica con la api desarrollada en el sprint 1, la duración de este sprint fue de 1 semana.

En la **Figura 13** se muestra que la comunicación con la api, se realizó utilizando API Fetch, es un estándar que utiliza promesas, es decir, devuelve un objeto con dos métodos, uno llamado then() que será invocado cuando se obtenga una respuesta y otro catch(), que se ejecutara solo si hay un error de red o de otro tipo.

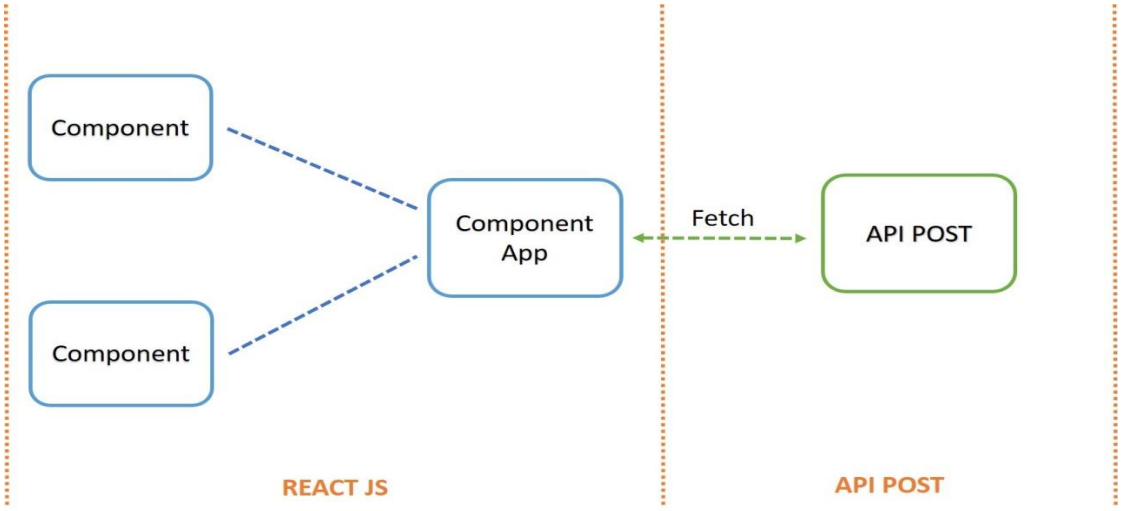


Figura 13. Arquitectura Front End – React js. Fuente: Adaptado de [88]

Una vez definido la arquitectura se inició el desarrollo de la aplicación web, donde cada día se cumplió con el scrum diario para hacer el seguimiento del sprint 2. Esta vez mientras se desarrollaba se hicieron los casos de prueba para realizar las pruebas respectivas de la aplicación. En la **Tabla 40**, se muestran los casos de prueba.

Tabla 40. Casos de prueba Sprint 2. Fuente: Propia.

ID Caso prueba	Título del caso de prueba	Precondiciones	Acción	Resultado esperado
TC_01	Buscar oración de un corpus	1. El servicio web debe estar desplegado 2. Navegar a la siguiente ruta: https://master.d2i78ejpt3vaot.amplifyapp.com/	1. Ubicarse en el campo de la frase 2. Seleccionar el corpus del campo desplegable 3. Digitar un texto a buscar 4. Seleccionar de la lista una frase coincidente	2. la aplicación muestra el corpus seleccionado en el campo 3. la aplicación muestra en una lista los textos coincidentes 4. la aplicación carga en el campo la frase seleccionada
TC_02	Ejecutar el proceso de etiquetado	1. El servicio web debe estar desplegado 2. Navegar a la siguiente ruta 3. Ejecutar el caso de TC_01	1. Clic en el campo de “Algoritmo” 2. Seleccionar un algoritmo 3. Clic en el botón ejecutar	1. La aplicación despliega una lista con la posibilidad de algoritmos 2. La aplicación habilita el botón de ejecutar 3. La aplicación muestra mensaje de “proceso de etiquetado ejecutando” 4. Una vez terminado el proceso, se muestra en pantalla las etiquetas de la frase y el resultado del etiquetado

Para cumplir con los criterios de aceptación de cada historia de usuario, se ejecutaron los casos de prueba, a continuación, se describen las pruebas realizadas:

- Para la HU5, se ejecutó el caso de prueba TC_01, donde se verifica la búsqueda de un texto y se muestra los resultados coincidentes, en la **Figura 14** se muestra cuando se realiza la búsqueda y en la **Figura 15** se muestra cuando se selecciona una frase.

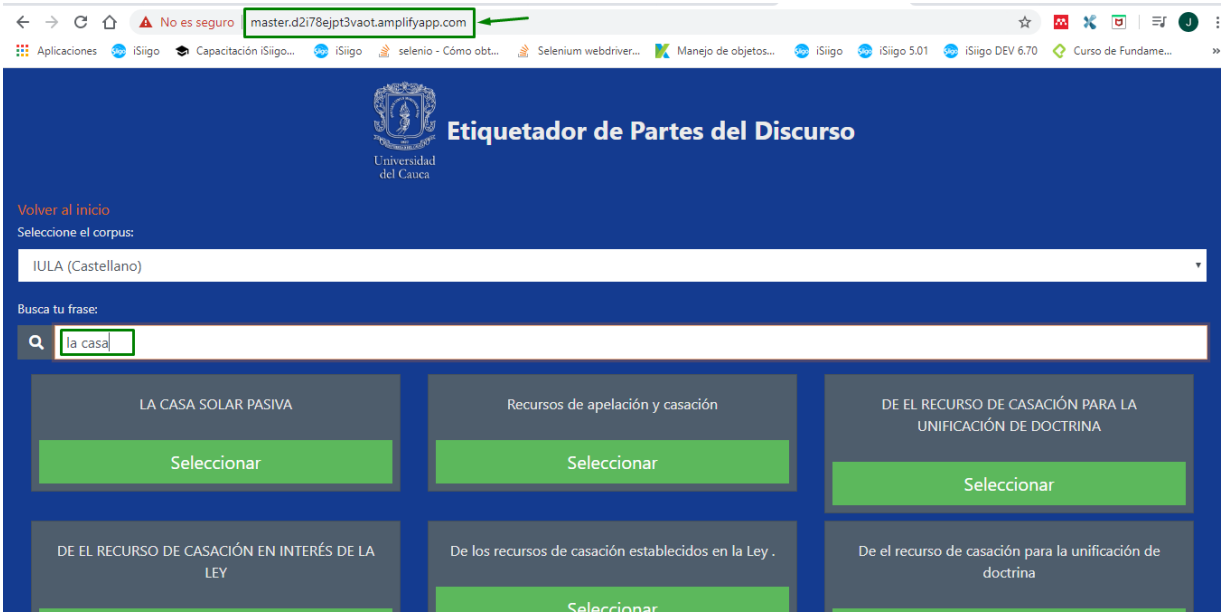


Figura 14. búsqueda de cadena en la app. Fuente: Propia

Figura 15. Selección de frase en la app. Fuente: Propia

- Para la HU6 y HU7, se ejecutó el caso de prueba TC_02, donde se verifica la selección de un algoritmo y ejecutar el etiquetado, en la **Figura 16** se muestra la selección de un algoritmo y en la **Figura 17** se muestra el resultado de la ejecución.

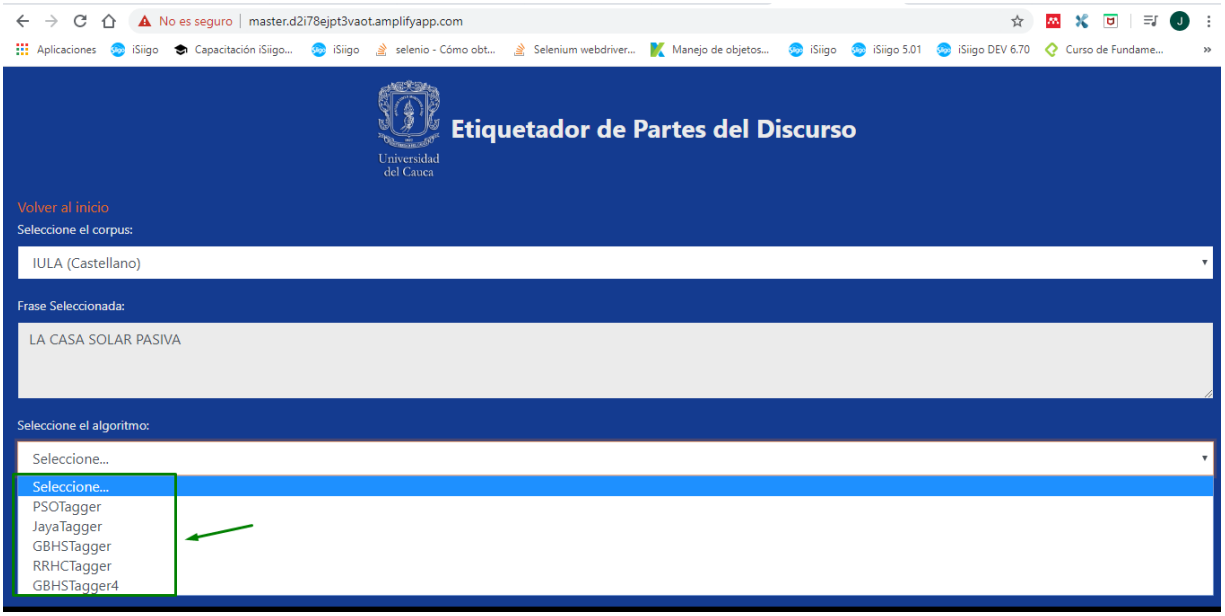


Figura 16. Selección de un algoritmo en la app. Fuente: Propia

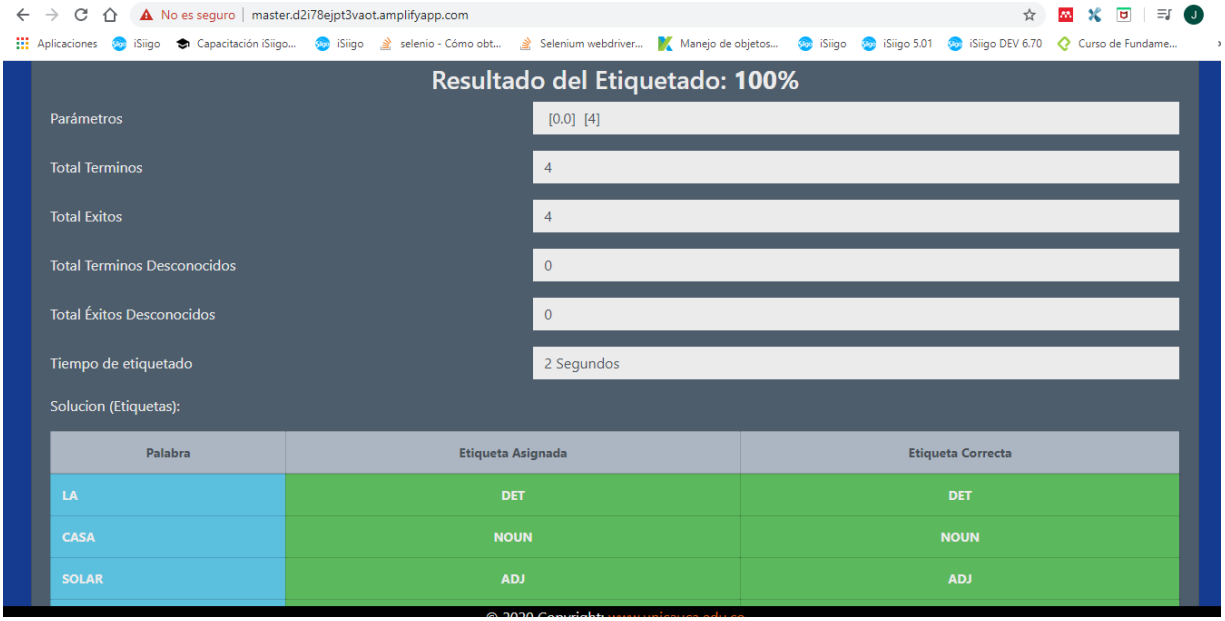


Figura 17. Resultado de etiquetado de un algoritmo en la app. Fuente: Propia

Realizadas las respectivas pruebas, se revisó el estado del sprint y el incremento, donde se cumplió con todo lo planificado. Por último, se realizó la retrospectiva del sprint 2, que fue presentada en un formulario, con el objetivo de documentar y que sirva como mejora continua del proceso, en la **Tabla 41**.

Tabla 41. Reunión de retrospectiva – sprint 2. Fuente: adaptado de [34]

¿Que salió bien en la iteración (aciertos)?	¿Qué no salió bien en la iteración?	¿Qué mejoras vamos a implementar en la próxima iteración? (recomendaciones de mejora continua)
Se cumplió con lo estimado El desarrollo y casos de prueba se hicieron simultáneamente	La ejecución de casos de prueba duró más de lo estimado	Hacer una tarea de integración del servicio web con la aplicación web

5.4 Publicación del Servicio web y Aplicación Web

El servicio web se encuentra alojado en una máquina virtual, en la url http://posttesis.southcentralus.cloudapp.azure.com/ y la aplicación web en un servicio app en la url https://master.d2i78ejpt3vaot.amplifyapp.com/. Debido a que el servicio recibe peticiones http al abrir la url de la app, hay que habilitar el contenido no seguro para el sitio. En el Anexo Digital 4 (shorturl.at/fgmuZ) se explica el proceso de despliegue de los dos artefactos y la comunicación final para habilitar el contenido no seguro para el sitio.

5.5 Síntesis

En el presente capitulo, se apreció el desarrollo de un servicio web y aplicación web utilizando la metodología ágil SCRUM como marco de trabajo, con el fin de utilizar una aplicación para mostrar el proceso de etiquetado y así se pueda seguir integrando en aplicaciones reales.

La aplicación permite seleccionar una frase de los corpus IULA, Brown y Nasa Yuwe, para luego con un determinado etiquetador ejecutar el proceso de etiquetado a dicha frase, devolviendo el resultado del etiquetado y las respectivas etiquetas de las palabras, pudiendo interactuar con los diferentes algoritmos presentados en esta investigación.

En el anexo digital 4 (shorturl.at/fgmuZ), se encuentra la aplicación Web, las instrucciones y videos para su publicación y en el link https://bit.ly/2WJCI4Z, se encuentra temporalmente ubicada la aplicación.

Capítulo 6

Conclusiones y trabajo futuro

En este capítulo se presentan los comentarios finales relacionados con el desarrollo de este trabajo. En la sección 6.1, se describen las conclusiones y en la sección 6.2, el trabajo futuro.

6.1 Conclusiones

A nivel de logros y aportes alcanzados en este trabajo se pueden resaltar así:

Este trabajo contó con dos corpus ya procesados como fueron el corpus Brown para el inglés y corpus Nasa Yuwe, los cuales habían sido utilizados en trabajos anteriores, pero en la revisión de literatura realizada, no se encontró el uso de corpus para Castellano en enfoques que utilizaran algoritmos metaheurísticos para el problema de etiquetado. Por tanto, en primera instancia, fue requerido investigar sobre corpus para Castellano y construir el dataset para esta lengua, obteniendo el dataset del corpus IULA, y su correspondiente mapeo al Conjunto de etiquetas Petrov.

En segunda instancia, se logró integrar en una sola base de datos los tres dataset para los corpus Brown, IULA y Nasa Yuwe.

En tercera instancia, este trabajo, elaboró la línea base para el Corpus IULA en relación con los algoritmos metaheurísticos azar, HS Tagger y el memético GBHS Tagger 2.

En cuarta instancia, este trabajo logró la adaptación de los algoritmos metaheurísticos Jaya [27], PSO [30] y GBHS [25] al problema de etiquetado, teniendo en cuenta las características propias de cada algoritmo y realizando el ajuste de parámetros requerido para cada algoritmo sobre cada corpus. Obteniendo resultados competitivos para el problema de etiquetado utilizando algoritmos metaheurísticos y pudiéndose apreciar que los mejores de precisión los alcanzó el algoritmo memético GBHS Tagger utilizando optimizador local.

En quinta instancia, se realizó un proceso exhaustivo de investigación en pro de modelar una nueva versión memética para el problema de etiquetado, como se presenta en el capítulo 4; logrando lo siguientes resultados:

- Adaptación de los algoritmos de estado simple Hill Climbing y Random Restart Hill Climbing al problema de etiquetado, incluyendo el respectivo ajuste de parámetros y obteniendo resultados competitivos para cada corpus.
- Para el corpus IULA se pudo obtener una mejora del algoritmo memético GBHS Tagger, basada en los contextos, la cual modificó la función objetivo, obteniendo los mejores resultados de precisión para este corpus.
- También se obtuvo otra versión memética del algoritmo GBHS Tagger al incluirle el algoritmo Hill Climbing, llamada GBHS Tagger 4, la cual obtiene competitivos para los tres corpus, incluso para el corpus Nasa Yuwe, es el que mejores resultados reporta.

Se logró realizar un prototipo software para etiquetado de oraciones de los tres corpus que utiliza algunos de los algoritmos adaptados y evaluados en este trabajo. Este prototipo utilizó SCRUM como metodología para su construcción, la cual facilitó todo el proceso de construcción del prototipo, así como su diseño de arquitectura.

A nivel de lo que pudo observar durante la realización de este trabajo, se puede resumir así:

El uso de algoritmo metaheurísticos en el problema de etiquetado permitió

- Mejorar la precisión de palabras desconocidas como se pudo apreciar en los experimentos realizados sobre la mejora presentada para el castellano GBHS2Pred2, en los resultados mostrados.
- La precisión en el etiquetado en todos los algoritmos presentados sobre los diferentes corpus fue competitiva, más específicamente el algoritmo Jaya con una precisión para el castellano de 96,8592, el cual es un valor muy cercano a la precisión del algoritmo GBHS Tagger 2.

El algoritmo GBHS superó en precisión a los otros algoritmos metaheurísticos Jaya y PSO, sin embargo, se destaca que obtuvieron muy buenos resultados, especialmente la adaptación del algoritmo Jaya, la cual es una metaheurística nueva y no había sido adaptada al problema de etiquetado, este algoritmo se caracteriza por ser muy sencillo de implementar y no requerir parámetros. Otro aspecto importante es la rapidez en que encuentra la solución.

El uso de algoritmos metaheurísticos en esta investigación demostró que son apropiados para el problema de etiquetado, como se pudo observar en los experimentos hechos sobre el corpus IULA, los cuales obtuvieron muy buenos resultados y se considera un buen aporte para el etiquetado de la lengua castellana.

El uso de un conjunto de etiquetas simplificado como el de [57], el cual fue implementado en todos los datasets de este trabajo, ayudó a mejorar el etiquetado dado que la tarea de reconocimiento de etiquetas es más fácil, en lugar de trabajar con un conjunto de etiquetas que contenga etiquetas similares.

El desarrollo de este trabajo involucró trabajar con metaheurísticas de estado simple, no obstante, se adaptó la metaheurística Restart Random Hill Climbing al problema de etiquetado, la cual utiliza una única solución y hace un balance entre exploración y explotación. Los experimentos demostraron que superó a varios algoritmos metaheurísticos poblacionales, convirtiéndose en el segundo mejor algoritmo de este trabajo para los tres corpus, esto nos anima en seguir buscando soluciones simples.

6.2 Trabajo futuro

El trabajo futuro se enfocará en los siguientes aspectos:

- Seguir utilizando algoritmos metaheurísticos para el etiquetado sobre otras lenguas tradicionales y no tradicionales.
- Buscar mejorar los algoritmos metaheurísticos Jaya, GBHS y RRHC, con otras técnicas de optimización, que sigan mejorando el etiquetado.
- Realizar el etiquetado para un corpus paralelo que contengan textos en dos idiomas, con el fin de unificar los resultados para los dos idiomas.
- Seguir construyendo herramientas que permitan enriquecer el etiquetado de partes del discurso, así como también el uso de las herramientas por parte de la comunidad en general, para así propagar la investigación en temas relacionados con el POST y algoritmos metaheurísticos.
- Desarrollar aplicaciones que puedan utilizar los etiquetadores desarrollados en este trabajo.

Bibliografía

[1] G. G. Chowdhury, “Natural language processing,” *Annu. Rev. Inf. Sci. Technol.*, vol. 37, no. 1, pp. 51–89, Jan. 2005, doi: 10.1002/aris.1440370103.

[2] A. Ekbal and S. Saha, “Simulated annealing based classifier ensemble techniques: Application to part of speech tagging,” *Inf. Fusion*, vol. 14, no. 3, pp. 288–300, 2013, doi: 10.1016/j.inffus.2012.06.002.

[3] K. Ravi and V. Ravi, “A survey on opinion mining and sentiment analysis: Tasks, approaches and applications,” *Knowledge-Based Syst.*, vol. 89, no. C, pp. 14–46, 2015, doi: 10.1016/j.knosys.2015.06.015.

[4] M. Bordoloi and S. K. Biswas, “Graph-Based Sentiment Analysis Model for E-Commerce Websites’ Data,” *Cogn. Informatics Soft Comput. Proceeding CISC 2017*, pp. 453–462, 2019, doi: 10.1007/978-981-13-0617-4.

[5] J. Ma, H. Liu, D. Huang, and W. Sheng, “An English part-of-speech tagger for machine translation in business domain,” *7th Int. Conf. Nat. Lang. Process. Knowl. Eng.*, pp. 183–189, 2011, doi: 10.1109/NLPKE.2011.6138191.

[6] S. Huet, G. Gravier, and P. Sébillot, “Morphosyntactic resources for automatic speech recognition,” *6th Int. Conf. Lang. Resour. Eval. Lr. 2008*, pp. 692–698, 2008.

[7] R. Karimpour *et al.*, “Improving persian information retrieval systems using stemming and part of speech tagging,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5706, pp. 89–96, 2009, doi: 10.1007/978-3-642-04447-2_10.

[8] T. Güngör, *Handbook of Natural Language Processing (second edition)*. 2011.

[9] D. Jurafsky and J. H. Martin, “Speech and Language Processing,” *Speech Lang. Process. An Introd. to Nat. Lang. Process. Comput. Linguist. Speech Recognit.*, vol. 21, pp. 151–176, 2009, doi: 10.1162/089120100750105975.

[10] E. Brill, “A simple rule-based part of speech tagger,” *Proc. third Conf. Appl. Nat. Lang. Process.*, 1992, doi: 10.3115/974499.974526.

[11] D. Q. Nguyen, D. Q. Nguyen, D. D. Pham, and S. B. Pham, “A robust transformation-based learning approach using ripple down rules for part-of-speech tagging,” *AI Commun.*, vol. 29, no. 3, pp. 409–422, 2016, doi: 10.3233/AIC-150698.

[12] L. Araujo, “Symbiosis of Evolutionary Techniques and Statistical Natural Language Processing,” *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 14–27, 2004, doi: 10.1109/TEVC.2003.818189.

[13] A. P. Silva, A. Silva, and I. Rodrigues, *Part-of-speech tagging using*

evolutionary computation, vol. 512.Spring. 2014.

[14] A. Paul, B. S. Purkayastha, and S. Sarkar, “Hidden Markov Model based Part of Speech Tagging for Nepali language,” *undefined*, 2015.

[15] A. Ratnaparkhi, “A Maximum Entropy Model for Part-Of-Speech Tagging,” *Proc. Conf. Empir. Methods Nat. Lang. Process. EMNLP-96*, pp. 133–142, 1996, doi: 10.1007/s00280-013-2178-x.

[16] T. Brants, “A Statistical Part-of-Speech Tagger,” vol. 5, pp. 1–7, 2000.

[17] D. Surendran and G.-A. Levow, “Dialog Act Tagging with Support Vector Machines and Hidden Markov Models,” *Interspeech 2006 9th Int. Conf. Spok. Lang. Process.*, pp. 1950–1953, 2006.

[18] H. Schmid, “Part-of-speech tagging with Neural Networks,” *Eur. J. Cancer Prev.*, vol. 27, no. 4, pp. 296–302, 1994, doi: 10.1097/CEJ.0000000000000354.

[19] L. Araujo, G. Luque, and E. Alba, “Metaheuristics for Natural Language Tagging,” pp. 889–900, 2004, doi: 10.1007/978-3-540-24854-5_90.

[20] W. N. Francis and H. Kucera, “Brown Corpus Manual,” 1979. [Online]. Available: http://clu.uni.no/icame/manuals/BROWN/INDEX.HTM#bc8. [Accessed: 03-Dec-2018].

[21] A. P. Silva, A. Silva, and I. Rodrigues, “BioPOS : Biologically Inspired Algorithms for POS Tagging,” *Proc. First Int. Work. Optim. Tech. Hum. Lang. Technol.*, vol. 2, no. December, pp. 1–16, 2013.

[22] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, “Building a large annotated corpus of English: the penn treebank,” *Comput. Linguist.*, vol. 19, no. 2, pp. 313--330, 1993, doi: 10.1162/coli.2010.36.1.36100.

[23] “Mac-Morpho.” [Online]. Available: http://nilc.icmc.usp.br/macmorpho/. [Accessed: 05-Dec-2018].

[24] R. Forsati and M. Shamsfard, “Novel harmony search-based algorithms for part-of-speech tagging,” *Knowl. Inf. Syst.*, vol. 42, no. 3, pp. 709–736, 2014, doi: 10.1007/s10115-013-0719-6.

[25] L. M. Sierra Martínez, C. A. Cobos, and J. C. Corrales, “Memetic algorithm based on global-best harmony search and hill climbing for part of speech tagging,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10682 LNAI, pp. 198–211, 2017, doi: 10.1007/978-3-319-71928-3_20.

[26] L. M. Sierra Martínez, C. A. Cobos, C. J. Muñoz Corrales, T. Curieux Rojas, E. Herrera-viedma, and D. H. Peluffo-ordóñez, “Building a Nasa Yuwe Language Corpus and Tagging with a Metaheuristic Approach,” *19th Int. Conf. Intell. Text*

Process. Comput. Linguist., vol. 22, no. 3, pp. 881–894, 2018, doi: 10.13053/CyS-22-3-3018.

[27] P. Singh and H. Chaudhary, “A Modified Jaya Algorithm for Mixed-Variable Optimization Problems,” *J. Intell. Syst.*, vol. 0, no. 0, pp. 1–21, 2018, doi: 10.1515/jisys-2018-0273.

[28] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, “Metaheuristic research: a comprehensive survey,” *Artif. Intell. Rev.*, no. January, pp. 1–43, 2018, doi: 10.1007/s10462-017-9605-z.

[29] W. G. M. Wolpert, David H., “No Free Lunch Theorems for Optimization,” *Encycl. GIS*, vol. 1, no. 1, pp. 240–240, 2008, doi: 10.1109/4235.585893.

[30] F. Neri, E. Mininno, and G. Iacca, “Compact particle swarm optimization,” *Inf. Sci. (Ny).*, vol. 239, pp. 96–121, 2013, doi: 10.1016/j.ins.2013.03.026.

[31] T. Rojas, “Desde arriba y por abajo construyendo el alfabeto nasa. La experiencia de la unificación del alfabeto de la lengua páez (nasa yuwe) en el Departamento del Cauca Colombia.,” *Construyendo El Alf.*, vol. 1, no. 1, p. 15, 2001.

[32] J. Brownlee, *Clever Algorithms*. 2011.

[33] S. Luke, *Essentials of Metaheuristics*. 2015.

[34] K. Schwaber and J. Sutherland, “La Guía de Scrum. La Guía Definitiva de Scrum: Las Reglas del Juego.,” p. 22, 2017.

[35] K. S. Pratt, “Design Patterns for Research Methods: Iterative Field Research,” *AAAI Spring Symp. Exp. Des. Real*, no. 1994, pp. 1–7, 2009.

[36] I. Project Management Institute, *Institute, A guide to the Project Management Body of Knowledge, 5 ed., Project Management Institute*. 2017.

[37] D. Cutting, J. Kupiec, J. Pedersen, and S. Penelope, “A Practical Part-of-Speech Tagger,” *Proc. Conf.*, pp. 133–140, 1992.

[38] R. Forsati, “Cooperation of Evolutionary and Statistical,” no. Aisp, pp. 550–555, 2012.

[39] I. Boussaïd, J. Lepagnot, and P. Siarry, “A survey on optimization metaheuristics,” *Inf. Sci. (Ny).*, vol. 237, pp. 82–117, Jul. 2013, doi: 10.1016/j.ins.2013.02.041.

[40] C. Cotta, “Una Visión General de los Algoritmos Meméticos.”

[41] S. Luke, *Essentials of Metaheuristics*. 2015.

[42] E. R. R. Kato, G. D. de A. Aranha, and R. H. Tsunaki, “A new approach to solve

the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing," *Comput. Ind. Eng.*, vol. 125, pp. 178–189, Nov. 2018, doi: 10.1016/j.cie.2018.08.022.

[43] Y. Zhang, S. Wang, and G. Ji, "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications," *Mathematical Problems in Engineering*, vol. 2015. Hindawi Limited, 2015, doi: 10.1155/2015/931256.

[44] E. R. R. Kato, G. D. de A. Aranha, and R. H. Tsunaki, "A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing," *Comput. Ind. Eng.*, vol. 125, pp. 178–189, Nov. 2018, doi: 10.1016/j.cie.2018.08.022.

[45] R. Venkata Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *Int. J. Ind. Eng. Comput.*, vol. 7, no. 1, pp. 19–34, Dec. 2016, doi: 10.5267/j.ijiec.2015.8.004.

[46] R. Kennedy, J., & Eberhart, "Particle swarm optimization," *Proc. Proc. IEEE Int. Conf. neural networks*, vol. Vol. 4, pp. 1942–1948, 1995.

[47] M. Sevkli and A. R. Guner, "A Continuos Particle Swarm Optimization Algorithm for Uncapacited Facility Location Problem," pp. 316–323, 2006.

[48] Q. Pan, M. F. Tasgetiren, and Y. Liang, "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem," vol. 35, pp. 2807–2839, 2008, doi: 10.1016/j.cor.2006.12.030.

[49] C. J. Liao, Chao-Tang Tseng, and P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Comput. Oper. Res.*, vol. 34, no. 10, pp. 3099–3111, Oct. 2007, doi: 10.1016/j.cor.2005.11.017.

[50] M. Clerc, "Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem," Springer, Berlin, Heidelberg, 2004, pp. 219–239.

[51] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Appl. Soft Comput. J.*, vol. 11, no. 4, pp. 3658–3670, 2011, doi: 10.1016/j.asoc.2011.01.037.

[52] A. Abunaser, I. A. Doush, N. Mansour, and S. Alshattnawi, "Underwater image enhancement using particle swarm optimization," *J. Intell. Syst.*, vol. 24, no. 1, pp. 99–115, Mar. 2015, doi: 10.1515/jisys-2014-0012.

[53] M. G. H. Omran and M. Mahdavi, "Global-best harmony search," *Appl. Math. Comput.*, vol. 198, no. 2, pp. 643–656, 2008, doi: 10.1016/j.amc.2007.09.004.

[54] A. Alhasan and A. T. Al-taani, "POS Tagging for Arabic Text Using Bee Colony Algorithm," *Procedia Comput. Sci.*, pp. 158–165, 2018, doi: 10.1016/j.procs.2018.10.471.

[55] M. El-Haj and R. Koulali, “KALIMAT a multipurpose Arabic Corpus,” *Second Work. Arab. Corpus Linguist.*, 2013.

[56] A. Al-Taani and S. A. Al-Rub, “A rule-based approach for tagging non-vocalized Arabic words,” *Int. Arab J. Inf. Technol.*, vol. 6, no. 3, pp. 320–328, 2009.

[57] S. Petrov, D. Das, and R. McDonald, “A Universal Part-of-Speech Tagset,” 2011, doi: 10.1038/hdy.2008.34.

[58] R. Forsati, M. Shamsfard, and P. Mojtahedpour, “An efficient meta heuristic algorithm for POS-tagging,” *Proc. - 5th Int. Multi-Conference Comput. Glob. Inf. Technol. ICCGI 2010*, pp. 93–98, 2010, doi: 10.1109/ICCGI.2010.42.

[59] E. Alba, G. Luque, and L. Araujo, “Natural language tagging with genetic algorithms,” *Inf. Process. Lett.*, vol. 100, no. 5, pp. 173–182, Dec. 2006, doi: 10.1016/J.IPL.2006.07.002.

[60] R. Forsati and M. Shamsfard, “Hybrid PoS-tagging: A cooperation of evolutionary and statistical approaches,” *Appl. Math. Model.*, vol. 38, no. 13, pp. 3193–3211, Jul. 2014, doi: 10.1016/J.APM.2013.11.047.

[61] A. P. Silva, A. Silva, and I. Rodrigues, “An Approach to the POS Tagging Problem Using Genetic Algorithms,” in *In Computation Intellingence*, 2012, pp. 3–17.

[62] A. P. Silva, A. Silva, and I. Rodrigues, “A New Approach to the POS Tagging Problem Using Evolutionary Computation,” *Proc. Recent Adv. Nat. Lang. Process.*, no. September, pp. 619–625, 2013.

[63] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, “A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA,” *IEEE Trans. Evol. Comput.*, vol. 12, no. 3, pp. 269–283, Jun. 2008, doi: 10.1109/TEVC.2007.900837.

[64] T. Curieux Rojas, “Esbozo gramatical de la lengua nasa (lengua páez),” *El Leng. en Colomb.*, vol. 1: Realida, no. Ed., Bogotá, Academia Colombiana de la Lengua e Instituto Caro y Cuervo, 2012.

[65] H. M. Alonso and D. Zeman, “Universal dependencies for the AnCora treebanks,” *Proces. Leng. Nat.*, vol. 57, pp. 91–98, 2016.

[66] Universal Dependencies, “Universal Dependencies.” [Online]. Available: http://universaldependencies.org/. [Accessed: 17-Dec-2018].

[67] J. Lavid, J. Arús, B. DeClerck, and V. Hoste, “Creation of a High-quality, Register-diversified Parallel (English-Spanish) Corpus for Linguistic and Computational Investigations,” *Procedia - Soc. Behav. Sci.*, vol. 198, pp. 249–256, Jul. 2015, doi: 10.1016/J.SBSPRO.2015.07.443.

[68] K. Toutanova, D. Klein, C. Manning, and Y. Singer, “Feautre-Rich Part-of-Speech Tagging with a Cyclic Dependency Network,” *Proc. 2003 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol.*, vol. 1, no. 4, pp. 173–180, 2003, doi: 10.1007/BF02379273.

[69] U. P. F. Institut Universitari de Lingüística Aplicada (IULA), “IULA Spanish LSP Treebank,” 2012.

[70] EAGLES, “ETIQUETAS EAGLES.” [Online]. Available: http://blade10.cs.upc.edu/freeling-old/doc/tagsets/tagset-es.html. [Accessed: 03-Dec-2018].

[71] I. Zeroual, A. Lakhouaja, and R. Belahbib, “Towards a standard Part of Speech tagset for the Arabic language,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 29, no. 2, pp. 171–178, 2017, doi: 10.1016/j.jksuci.2017.01.006.

[72] H. Schmid, “TreeTagger - a language independent part-of-speech tagger,” 1994. [Online]. Available: http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/treetagger.en.html. [Accessed: 12-Dec-2018].

[73] B. Kabashi and T. Proisl, “A Proposal for a Part-of-Speech Tagset for the Albanian Language,” *Proc. Tenth Int. Conf. Lang. Resour. Eval. (LREC 2016)*, pp. 4305–4310, 2016.

[74] M.-C. De Marneffe, B. MacCartney, and C. D. Manning, “Generating Typed Dependency Parses from Phrase Structure Parses,” *Proc. Lr.*, 2006, doi: 10.1145/1391729.1391731.

[75] D. Zeman, “Reusable Tagset Conversion Using Tagset Drivers,” *Proc. Lr.*, 2008.

[76] J. Lafferty, A. Mccallum, and F. Pereira, “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data Part of the Numerical Analysis and Scientific Computing Commons Recommended Citation "Conditional Random Fields: Probabilistic Models for Segmenting and Labelin,” *Proc. ICML*, vol. 2001, no. June, pp. 282–289, 2001.

[77] J. Tobar and M. Solano, “Dataset IULA,” 2020. .

[78] “slavpetrov/universal-pos-tags: Automatically exported from code.google.com/p/universal-pos-tags.” [Online]. Available: https://github.com/slavpetrov/universal-pos-tags. [Accessed: 27-Jun-2019].

[79] J. Alcalá-Fdez *et al.*, “KEEL: A software tool to assess evolutionary algorithms for data mining problems,” *Soft Comput.*, vol. 13, no. 3, pp. 307–318, 2009, doi: 10.1007/s00500-008-0323-y.

[80] S. Callander, “Searching and learning by trial and error,” *Am. Econ. Rev.*, vol.

101, no. 6, pp. 2277–2308, 2011, doi: 10.1257/aer.101.6.2277.

[81] A. K. Mishra and D. Shrivastava, “A discrete Jaya algorithm for permutation flow-shop scheduling problem,” vol. 11, pp. 1–14, 2020, doi: 10.5267/j.ijiec.2019.12.001.

[82] S. O. Degertekin, L. Lamberti, and I. B. Ugur, “Sizing, layout and topology design optimization of truss structures using the Jaya algorithm,” *Appl. Soft Comput. J.*, vol. 70, pp. 903–928, Sep. 2018, doi: 10.1016/j.asoc.2017.10.001.

[83] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan, “Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm,” *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1944–1955, 2019, doi: 10.1109/TCYB.2018.2817240.

[84] M. A. Al-Betar, “β-Hill climbing: an exploratory local search,” *Neural Comput. Appl.*, vol. 28, no. 1, pp. 153–168, Dec. 2017, doi: 10.1007/s00521-016-2328-2.

[85] C. Laguna, “Inferencia Paramétrica: Relación entre dos variables cualitativasy cuantitativas.,” *Inst. Aragon. ciencias la salud*, vol. 8, no. 2, p. 2,10, 2016.

[86] M. Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. 2011.

[87] “Insomnia REST Client.” [Online]. Available: https://insomnia.rest/. [Accessed: 31-Mar-2020].

[88] J. Walke, “React – Una biblioteca de JavaScript para construir interfaces de usuario.” [Online]. Available: https://es.reactjs.org/. [Accessed: 05-Apr-2020].

[89] L. M. Sierra, C. Cobos, and J. C. Corrales, “Continuous Optimization Based on a Hybridization of Differential Evolution with K-means,” *Bazzan A., Pichara K. Adv. Artif. Intell.*, vol. 8864, pp. 381–392, Nov. 2014, doi: 10.1007/978-3-319-12027-0_31.

[90] J. J. Tobar, M. A. Solano, L. M. Sierra, and C. A. Cobos, “Etiquetado de partes del discurso sobre un corpus en castellano basado en metaheurísticas,” *Rev. Ibérica Sist. e Tecnol. Inf.*, p. 15, 2020.

[91] D. Moussallem, M. Wauer, and A. N. Ngomo, “Web Semantics : Science , Services and Agents on the World Wide Web Machine Translation using Semantic Web Technologies : A Survey,” *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 51, pp. 1–19, 2018, doi: 10.1016/j.websem.2018.07.001.

[92] C. Garcia-Martinez, F. J. Rodriguez, and M. Lozano, “Ánalisis Empirico del No Free Lunch en Problemas Binarios Reales.” .

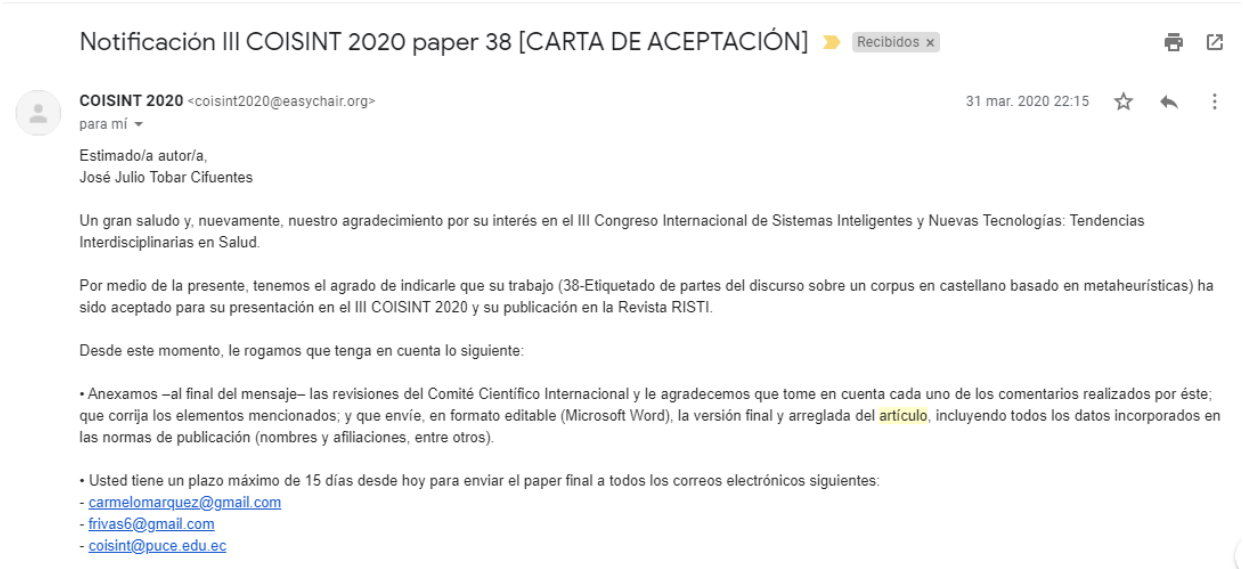
Anexo 1

Articulo aceptado en el III COISINT 2020

1.1 Congreso III COISINT 2020

III Congreso Internacional de Sistemas inteligentes y Nuevas tecnologías: Tendencias interdisciplinarias en salud. Publicaciones de artículos en la revista RISTI. Este articulo presenta la construcción del dataset IULA y la línea base para el castellano, además se presenta una mejora al algoritmo memético GBHS Tagger2. A continuación se presenta la carta de aceptación del artículo en el congreso.

Carta de aceptación



Notificación III COISINT 2020 paper 38 [CARTA DE ACEPTACIÓN] Recibidos ×

COISINT 2020 <coisint2020@easychair.org> 31 mar. 2020 22:15
para mí

Estimado/a autor/a,
José Julio Tobar Cifuentes

Un gran saludo y, nuevamente, nuestro agradecimiento por su interés en el III Congreso Internacional de Sistemas Inteligentes y Nuevas Tecnologías: Tendencias Interdisciplinarias en Salud.

Por medio de la presente, tenemos el agrado de indicarle que su trabajo (38-Etiquetado de partes del discurso sobre un corpus en castellano basado en metaheurísticas) ha sido aceptado para su presentación en el III COISINT 2020 y su publicación en la Revista RISTI.

Desde este momento, le rogamos que tenga en cuenta lo siguiente:

• Anexamos –al final del mensaje– las revisiones del Comité Científico Internacional y le agradecemos que tome en cuenta cada uno de los comentarios realizados por éste; que corrija los elementos mencionados; y que envíe, en formato editable (Microsoft Word), la versión final y arreglada del artículo, incluyendo todos los datos incorporados en las normas de publicación (nombres y afiliaciones, entre otros).

• Usted tiene un plazo máximo de 15 días desde hoy para enviar el paper final a todos los correos electrónicos siguientes:
- carmelomarquez@gmail.com
- frivas6@gmail.com
- coisint@puce.edu.ec

• El artículo, en formato editable, debe adaptarse con rigor a las normas y al formato de la revista. Este formato puede descargarlo en la página web del III COISINT 2020: http://coisint.puce.edu.ec. No olvide que no puede sobrepasar las quince (15) páginas.

• Asimismo, solicitamos que, tanto el asunto del correo enviado como el nombre del archivo adjunto, se conformen a través del número de paper asignado en Easychair. En su caso: "COISINT2020_paper_38".

• Por último, cabe señalar que, para publicar, es obligatorio realizar el registro y el pago en el Congreso. Sobre esta cuestión, le enviaremos próximamente las debidas instrucciones.

Le enviamos nuestras sinceras disculpas por no haber podido cumplir rigurosamente con las fechas establecidas desde hace meses. Atendemos a su máxima comprensión por la coyuntura global de crisis en torno al COVID-19.

Deseándole el mayor de los éxitos en todas sus actividades y en espera del envío de la versión final del artículo, enviamos un cordial saludo.

COMITÉ EDITORIAL
III COISINT 2020

SUBMISSION: 38
TITLE: Etiquetado de partes del discurso sobre un corpus en castellano basado en metaheurísticas

Revisión 1

---------- REVIEW 1 ----------
SUBMISSION: 38
TITLE: Etiquetado de partes del discurso sobre un corpus en castellano basado en metaheurísticas
AUTHORS: José Julio Tobar Cifuentes, Miguel Alexis Solano Jiménez, Luz Marina Sierra and Carlos Cobos

---------- Evaluación global // Overall evaluation ----------
SCORE: 2 (aceptación / accept)
----- TEXT:
Corresponde a un trabajo de gran pertinencia, particularmente en el contexto de aplicaciones de PLN.
El trabajo ha sido adecuadamente estructurado en seis secciones, partiendo de una introducción, trabajos relacionados, donde se presenta un panorama actualizado sobre el tema del etiquetamiento, la metodología utilizada, la construcción del conjunto de datos, la propuesta de mejora al algoritmo memético GBHS Tagger 2, los resultados obtenidos y las conclusiones.
La redacción del texto puede ser mejorada, específicamente en cuanto al uso de signos de puntuación, el uso de conectores apropiados en algunas oraciones y la manera de referenciar algunos trabajos.

Revisión 2

---------- REVIEW 2 ----------
SUBMISSION: 38
TITLE: Etiquetado de partes del discurso sobre un corpus en castellano basado en metaheurísticas
AUTHORS: José Julio Tobar Cifuentes, Miguel Alexis Solano Jiménez, Luz Marina Sierra and Carlos Cobos

---------- Evaluación global // Overall evaluation ----------
SCORE: 2 (aceptación / accept)
----- TEXT:
Los autores proponen la construcción de un dataset en castellano y la comparación de varios algoritmos metaheurísticos. Hacen una buena revisión del estado del arte quedando bien justificada la investigación. Es un trabajo bien escrito y bien argumentado.

Sin duda es aceptado para su publicación

Sin embargo, tengo algunas pequeñas correcciones:
1. En el Resumen
   • Corregir la palabra artículo

"... etiquetado de palabras desconocidas. Este articulo presenta la construcción de.."

por

".. etiquetado de palabras desconocidas. Este artículo presenta la construcción de.."

2. En la Introducción
   • Corregir las palabras: traducción y reconocimiento. Están en mayúsculas.

"...de polaridad (Bordoloi & Biswas, 2019); Traducción de textos,....textos (Ma, Liu, Huang, & Sheng, 2011); Reconocimiento de voz...."

3.En la subseccion: Construcción de dataset IULA
   • Corregir,la palabra forma, en
   "...estos campos son ID, Forma de la palabra, etiquetado POS de grano fino y de grano grueso, entre otros. Teniendo en cuent..."

Es todo.

1.2 Contenido del articulo

Etiquetado de partes del discurso sobre un corpus en castellano basado en metaheurísticas

José Julio Tobar C, Miguel Alexis Solano J, Luz Marina Sierra-M, Carlos Alberto Cobos L.

Resumen: El etiquetado de partes del discurso es una de las tareas más importantes en el preprocesamiento del lenguaje natural y tiene usos en el análisis de sentimientos, traducción de texto, reconocimiento de voz y recuperación de información, entre otros. Esta tarea se enfrenta a tres retos principales relacionados con la ambigüedad de las palabras, el tamaño del conjunto de etiquetas y el etiquetado de palabras desconocidas. Este artículo presenta la construcción de un dataset en castellano y la comparación de varios algoritmos metaheurísticos del estado del arte sobre el corpus en castellano, incluido un algoritmo memético mejorado que maneja diferentes contextos de las palabras, lo que le permite obtener un mejor desempeño.

Palabras-clave: Algoritmos metaheurísticos; Algoritmo memético; Corpus etiquetado IULA; Identificación de partes del discurso; Mejor búsqueda armónica global.

1. Introducción

El etiquetado de partes del discurso (Part-of-Speech tagging, POST) es un área del procesamiento de lenguaje natural (PLN) que se encarga de asignar a cada palabra de una oración sus posibles categorías léxicas, de acuerdo a su contexto [8], y se constituye como una tarea importante en el preprocesamiento de la mayoría de las aplicaciones de PLN. Algunas de las aplicaciones son: análisis de sentimientos, donde el etiquetado reconoce diferentes partes de un texto y ayuda en la determinación de polaridad [4]; traducción de textos, donde el etiquetado es crucial para una traducción de alta calidad, dado que proporciona características importantes de la fragmentación y el análisis de los textos [5]; reconocimiento de voz, donde el etiquetado es un paso preliminar que permite explotar las transcripciones de audio, aportando conocimiento lingüístico, como por ejemplo, información morfosintáctica, que genera mejores transcripciones [6]; y tareas de recuperación de información, donde el etiquetado ayuda en el post-procesamiento, dado que las palabras etiquetadas como sustantivos, pueden clarificar el rol de cada término en consultas y documentos [7].

El etiquetado se convierte en una tarea compleja, al enfrentar principalmente tres grandes retos [8], [9], a saber: 1) La ambigüedad de palabras, que puede ocasionar un etiquetado incorrecto, dadas las múltiples etiquetas que puede tener cada palabra; 2) El tamaño del conjunto de etiquetas, un conjunto de etiquetas muy especializado puede llegar a dificultar la distinción entre varias etiquetas similares, y un conjunto de etiquetas muy general, puede ocasionar pérdida de conocimiento morfológico y morfosintáctico ocasionando un etiquetado incorrecto; y 3) El etiquetado de palabras desconocidas, es decir, palabras que no se encuentran en los datos de entrenamiento, o sobre las cuales no se tienen previamente definidas reglas para su etiquetado, por tanto, se dificulta asignar automáticamente la etiqueta correcta.

El problema de etiquetado se aborda tradicionalmente con dos enfoques: 1) Los basados en reglas [10] y 2) Los basados en información estadística. Entre las técnicas más reconocidas se incluyen los modelos ocultos de Markov [14], los modelos de máxima entropía [15] los trigramas [16], las máquinas de soporte vectorial [17], las redes neuronales [18], entre otros.

Actualmente, existen nuevas propuestas que abordan el problema de etiquetado utilizando algoritmos metaheurísticos, tanto desde una perspectiva estadística como basada en reglas. Estas propuestas metaheurísticas han obtenido un desempeño sobresaliente en comparación con otras técnicas, como se puede apreciar en: [19] que utilizan un algoritmo genético, recocido simulado y el algoritmo CHC, para el problema de etiquetado; [21], que abordan el etiquetado como un problema de optimización

usando el algoritmo de optimización por enjambre de partículas (Particle Swarm Optimization, PSO). [24] que utilizan la metaheurística de la Búsqueda Armónica (Harmony Search, HS); entre otros trabajos. Estos enfoques metaheurísticos se aplican sobre corpus etiquetados de lenguas como el inglés y otras lenguas no tradicionales como: el árabe, bengalí (Bangladés), hindi (India), Telugu (India) y Nasa Yuwe (lengua indígena de Colombia).

En esta investigación se aborda el problema de etiquetado desde la perspectiva de los algoritmos metaheurísticos, con el objetivo de evaluar el desempeño de las metaheurísticas (HSTagger, HSTagger2 y HSTagger3) y el algoritmo memético propuesto en [25], sobre el corpus etiquetado IULA (Institut de Lingüstica Aplicada) del castellano. Estos algoritmos previamente se han utilizado sobre los corpus Brown del inglés [20] y del Nasa Yuwe [26]. Cabe aclarar, que no se ha encontrado referencia sobre la aplicación de algoritmos metaheurísticos de etiquetado sobre un corpus en castellano. Adicionalmente, en este trabajo se presenta, una mejora al algoritmo memético propuesto en [25] que involucra un cambio en el contexto (es decir, para etiquetar una palabra en la oración, se tienen en cuenta la(s) etiqueta(s) de la(s) palabra(s) predecesora(a) o sucesora(s) [24] de la palabra a ser etiquetada.

El resto del artículo está organizado de la siguiente manera: La sección 2 presenta los trabajos más relevantes sobre el problema de etiquetado desde una perspectiva metaheurística, los conjuntos de etiquetas y algunos de los corpus utilizados; La sección 3 resume el proceso metodológico seguido; La sección 4 presenta el procesamiento del corpus IULA; La sección 5 muestra la mejora realizada al algoritmo memético propuesto en [25]; La sección 6 muestra los experimentos realizados con los algoritmos metaheurísticos de la línea base y la mejora propuesta; y finalmente, en la sección 7 se presentan las conclusiones y algunas ideas de trabajos futuros.

2. Trabajos Relacionados

Una vez realizada la revisión de la literatura en las diferentes bases de datos científicas, se encontraron los siguientes trabajos relevantes sobre algoritmos metaheurísticos aplicados al problema de etiquetado, corpus etiquetados y conjuntos de etiquetas:

En 2018, [54] presentan un etiquetador utilizando el algoritmo de optimización por colonia de abejas (BCO) para la lengua árabe. En 2017, [25] presentan un etiquetador basado en Global-Best Harmony Search (GBHS) que incluye conocimiento del problema mediante una estrategia de optimización local basada en el algoritmo Hill Climbing [32], evaluado sobre una lengua tradicional (Inglés) y una no tradicional (Nasa

Yuwe). En 2015, [24] presentaron dos mejoras del etiquetador HSTAGger [60] que incrementan la eficiencia de búsqueda y mejoran la selección de nuevas soluciones para la memoria armónica, esta propuesta fue evaluada sobre el corpus Brown [20] y PennTreenbank [22]. En 2013, [62] dividen el problema de etiquetado en dos tareas: una de aprendizaje y otra de optimización, que ayuda al descubrimiento de reglas usando la computación evolutiva, utilizando dos algoritmos, un algoritmo genético (GA-Tagger) y un PSO binario (PSO-Tagger), que fue evaluado sobre el corpus Brown.

En 2016, [65] presentan el corpus UD Spanish Ancora, que contiene 17.680 oraciones y 547.682 tokens, además hacen la conversión del conjunto de etiquetas a dependencias universales [66] y consta de 17 etiquetas. En 2015, [67] presentan un corpus paralelo Inglés-Castellano de alta calidad que se compone de textos originales en los dos idiomas, el corpus puede ser utilizado en varias disciplinas. En 2012, el IULA [69] presentó el corpus IULA Spanish LSP Treebank que consta de 40.000 oraciones de la lengua castellana, tomadas del IULA technical corpus. Las etiquetas se basan en el conjunto de etiquetas ***EAGLES*** [70] para el castellano, el cual contiene 577 etiquetas. En 1993, [22] presentaron el corpus PennTreebank, que consta de 4.5 millones de palabras del Inglés americano. Este corpus ha sido anotado con información de partes del discurso, el conjunto de etiquetas está conformado por 36 etiquetas de POS y otras 12 etiquetas de puntuación y símbolos de moneda. En 1979, [20] presentaron el corpus Brown, que es una recopilación de 500 archivos de texto plano en Inglés. El conjunto de etiquetas está conformado por 87 categorías que se subdividen en un total de 472 etiquetas.

En 2017, [71] diseñaron un conjunto de etiquetas para la lengua arábica, que establece niveles jerárquicos de las categorías de las palabras, estos niveles jerárquicos permiten una expansión más fácil y mejoran el etiquetado. En 2016, [73] presentaron un conjunto de etiquetas de grano fino para el lenguaje albanés, el cual sigue un enfoque tradicional para las categorías de palabras. En 2014, más de 200 colaboradores presentaron un proyecto abierto llamado dependencias universales [66], el cual es un conjunto de anotación para muchos lenguajes, que está basado en las dependencias de Stanford [74], etiquetas universales [57], y el interlingua Interset para los conjuntos de etiquetas morfosintácticas [75]. En 2012, [57] presentan un conjunto de 12 etiquetas comunes para 25 lenguajes (a partir de 25 conjuntos de etiquetas) provenientes de trabajos previos buscando mejorar la precisión de los etiquetadores a través de varios lenguajes. Esta propuesta ha sido muy aceptada y utilizada en diferentes trabajos de identificación de partes del discurso.

Después de hacer la revisión del estado del arte se pudo encontrar que: 1) el problema de etiquetado está siendo cada vez más explorado desde diferentes enfoques. 2) El uso de algoritmos metaheurísticos en este problema, ha ido tomando fuerza en los últimos años (2010-2019). 3) El idioma inglés y el conjunto de etiquetas Universal [57] son los más utilizados para la evaluación de desempeño de los etiquetadores.

Adicionalmente, de la revisión se puede inferir que: 1) Los etiquetadores basados en algoritmos metaheurísticos obtienen mejores resultados o similares a los alcanzados con los enfoques tradicionales o híbridos, tanto en etiquetado de lenguas tradicionales como en lenguas no tradicionales. Cabe resaltar que solo existe una propuesta que utiliza un enfoque memético para el problema de etiquetado; 2) No todos los corpus revisados están mapeados al conjunto de etiquetas universal Petrov [57]; y 3) Muchos de los corpus presentados tienen su propio conjunto de etiquetas, el cual incluye mucha información de grano fino.

3. Metodología

En esta investigación, se utilizó el Patrón de Investigación Iterativa (Iterative Research Pattern, IRP) propuesto por [35]. Para esta investigación se realizaron dos ciclos. La Tabla 1, presenta un resumen de cada una de las actividades realizadas.

4. Construcción del dataset IULA

El corpus IULA [69] original se encuentra almacenado en un archivo con formato CONLL, donde cada línea representa un sólo término con una serie de 10 campos, separados por tabulaciones, alguno de estos campos son ID, Forma de la palabra, etiquetado POS de grano fino y de grano grueso, entre otros. Teniendo en cuenta este formato, se realizó el procesamiento para configurarlo como un dataset adecuado para la configuración de los experimentos con los algoritmos POST. En primera instancia, se realizó un modelo de base de datos, que permitiera a cada algoritmo consultar de manera eficiente la información del corpus, como: términos, frases, conjunto de etiquetas, tablas estadísticas, procedimientos almacenados y el folder de la frase para hacer validación cruzada. La Figura 18, presenta el modelo de base de datos construido, que incluye las tablas y sus respectivas relaciones entre frases, términos, etiquetas propias de los términos del corpus y su correspondiente etiqueta Universal.

Tabla 42 - Actividades definidas en cada ciclo. Fuente: Elaboración propia

Ciclo	Observación	Identificación	Desarrollo	Pruebas
Ciclo 1	Revisión de metaheurísticas e investigación de	Selección del corpus y mapeo de etiquetas al	Diseño e implementación de la base de datos para el corpus IULA.	Análisis y discusión de resultados de

	corpus en castellano	etiquetado universal	Configuración y ejecución del experimento con el algoritmo memético propuesto en (Sierra Martínez et al., 2017), y otros algoritmos metaheurísticos utilizados sobre el dataset IULA	precisión obtenidos
Ciclo 2	Exploración de ventanas contextuales para el castellano.	Definición de los contextos implementados.	Propuesta e implementación de una mejora al mejor algoritmo del ciclo 1 basado en los contextos y ejecución de los experimentos y selección de la mejor versión preliminar del algoritmo.	Análisis de resultados y emisión de conclusiones

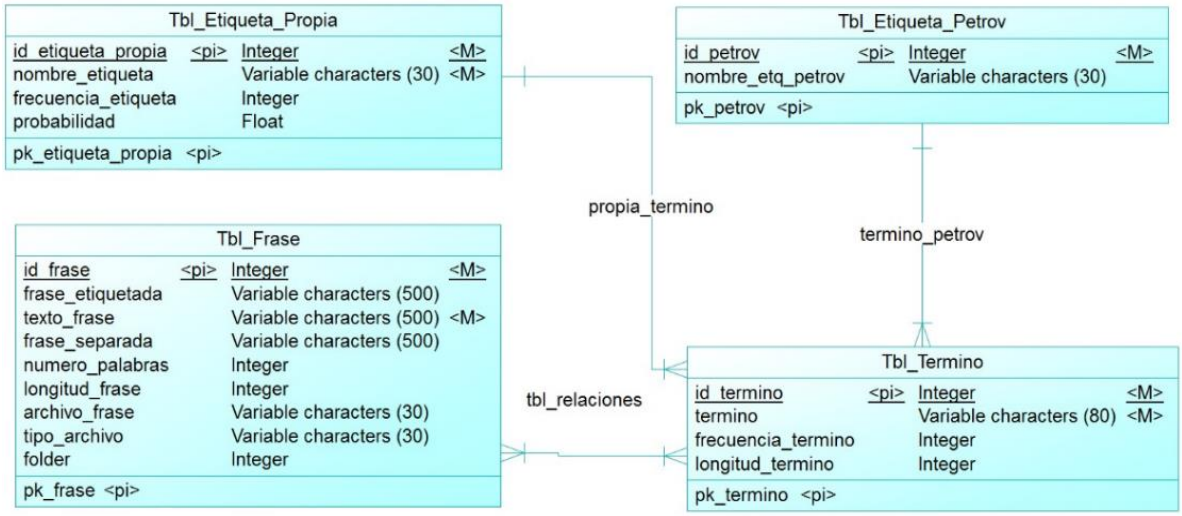


Figura 18 – Modelo Conceptual de la Base de Datos

En segunda instancia, y partiendo del modelo conceptual presentado en la Figura 1, se generó el modelo físico de la base de datos para el motor SQL Server 2017. En [77] se encuentran los scripts de la base de datos que incluyen la estructura y los datos. En tercera instancia, se procedió a cargar la información en la base de datos construida mediante un algoritmo en Python, que incluye la lectura del archivo del Corpus IULA, toma el término con su etiqueta y realiza la equivalencia con la etiqueta EAGLES [70] del término con su correspondiente etiqueta Universal [57] según el archivo de equivalencias “es-eagles.map” [78]. En la Tabla 2, se presenta un fragmento de la equivalencia entre etiquetas. Finalmente, con la lectura de estos términos se organizó cada frase hasta completar la totalidad de términos y frases del corpus en la base de datos.

Tabla 43 - Equivalencia de etiquetas EAGLES a PETROV

Id	Etiq_IULA	Etiq_Petrov
1	AQ0MS0	ADJ
182	SPS00	ADP
184	RG	ADV
186	CC	CONJ
188	DD0MS0	DET
235	NCMS000	NOUN

Id	Etiq_IULA	Etiq_Petrov
259	Z	NUM
265	PP1CSN00	PRON
363	VMIP2P0	VERB
555	W	X
557	Fat	.

Como resultado del procesamiento del corpus, se logró consolidar una base de datos con 42.079 frases y 37.763 términos. Dada la cantidad de registros y la cantidad de consultas que debe atender, se llevó a cabo un proceso de indexación sobre las tablas, para las llaves con mayor número de relaciones.

Una vez se completó el procesamiento del corpus IULA, se procedió a realizar los experimentos con los algoritmos GBHS, GBHS2, GBHS3, HSTagger, HSTagger2 y HSTagger3 y una búsqueda aleatoria (Azar), utilizando validación cruzada de cinco folders. Los resultados de estos experimentos se presentan en la sección 6.

Como resultado de los experimentos se pudo apreciar que el algoritmo memético GBHS Tagger 2, utilizando el optimizador local con probabilidades de 0.3, 0.5 y 0.7 logró obtener los mejores resultados para el corpus IULA, con una precisión de 97.6051%. Previamente con el Corpus Brown del inglés [20] en [25], se obtuvo una precisión de 95.1959% y en Nasa Yuwe [26] una precisión de 66.5787%.

Por tanto, el algoritmo memético GBHS Tagger 2, utilizando el optimizador local, fue seleccionado para realizarle modificaciones en pro de mejorar su desempeño en el problema de etiquetado. A continuación, se presenta el proceso seguido y la mejora propuesta para este algoritmo.

5. Mejora propuesta al algoritmo memético GBHS Tagger 2

El contexto utilizado en [25] para el etiquetado con el algoritmo memético GBHS2Tagger fue un trigrama, es decir, al momento de asignar la etiqueta de la palabra se tenía en cuenta la etiqueta de la palabra predecesora y la de la palabra sucesora, como se muestra en la Figura 19a.

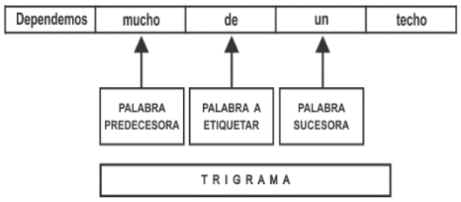


Figura 19a - Contexto del Trigrama

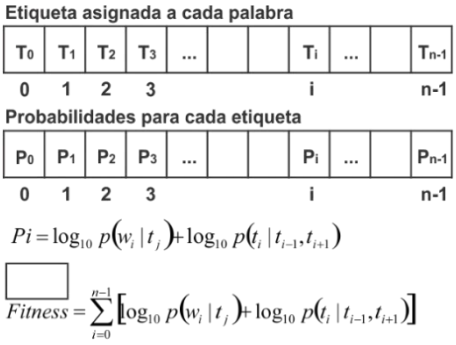


Figura 20a - Representación de la solución

La representación de la solución fue organizada como se muestra en la Figura 19b, así: el primer vector contiene las etiquetas asignadas a cada palabra de la frase a etiquetar, donde T_0, representa la etiqueta de la primera palabra y así sucesivamente para todas las palabras de la frase; el segundo vector contiene cada una de las probabilidades acumuladas para cada etiqueta, representadas en la Figura 20b como los P_i; finalmente, se tiene otro campo que almacena el valor de la función fitness, que se calcula como la sumatoria de los logaritmos de los P_i [25].

Teniendo en cuenta esa representación, las mejoras propuestas en la presente investigación, se enfocaron en revisar el desempeño del algoritmo memético GBHS Tagger 2 con diferentes contextos; se seleccionaron 4 contextos que mantienen la representación de la solución de la Figura 2b, cambiando la forma en que se calcula los P_i del segundo vector y la función fitness, así:

- **Contexto 1,** tiene en cuenta un tetragrama, conformado por dos palabras predecesoras, la palabra a etiquetar y una sucesora; realizando el cálculo de los P_i utilizando la Ecuación 5.

$$P_i = log_{10}P(w_i|t_i) + log_{10}P(t_i|\boldsymbol{t_{i-2}}, \boldsymbol{t_{i-1}}, \boldsymbol{t_{i+1}})$$

Ecuación 12 - Probabilidad del contexto 1

- **Contexto 2**, también se representa con un tetragrama conformado por: una palabra predecesora, la palabra a etiquetar y dos sucesoras; realizando el cálculo de los P_i utilizando la Ecuación 6.

$$P_i = log_{10}P(w_i|t_i) + log_{10}P(t_i|\boldsymbol{t_{i-1}}, \boldsymbol{t_{i+1}}, \boldsymbol{t_{i+2}})$$

Ecuación 13 - Probabilidad del contexto 2

- **Contexto 3**, es un trigrama organizado así: la palabra a etiquetar y dos sucesoras; realizando el cálculo de los P_i utilizando la Ecuación 7:

$$P_i = log_{10}P(w_i|t_i) + log_{10}P(t_i|\boldsymbol{t_{i+1}}, \boldsymbol{t_{i+2}})$$

Ecuación 14 - Probabilidad del contexto 3

- **Contexto 4**, también un trigrama organizado así: dos palabras predecesoras y la palabra a etiquetar; realizando el cálculo de los P_i utilizando la Ecuación 15.

$$P_i = log_{10}P(w_i|t_i) + log_{10}P(t_i|\boldsymbol{t_{i-2}}, \boldsymbol{t_{i-1}})$$

Ecuación 15 - Probabilidad del contexto 4

- **Contexto 5**, hace referencia al trigrama original del algoritmo, es decir, una palabra predecesora, la palabra a etiquetar y una sucesora. El cálculo de los P_i se hace como se presenta en la Figura 2b.

Seguidamente a la definición de los contextos, se configuraron experimentos preliminares que involucraron la construcción de un dataset con 5.000 frases, tomadas de forma aleatoria del dataset descrito en la sección 4. Los experimentos se ejecutaron utilizando validación cruzada de 5 folders, por ejemplo, si las oraciones del folder 1 fueron tomadas como datos de prueba, las oraciones de entrenamiento son tomadas desde el folder 2 a 5, y así sucesivamente para todos los folders. Cada folder con 1.000 oraciones de prueba y un promedio de palabras superior a 4.000, de las cuales el 35% o 36% eran desconocidas. Cada contexto se implementó e identificó en el algoritmo GBHS2Tagger así: GBHS2Pred2Suc, para el Contexto 1, GBHS2PredSuc2 Contexto 2, GBHS2Suc2 Contexto 3, GBHS2Pred2 Contexto 4, y finalmente, GBHS2PredSuc Contexto 5. En la Tabla 3, se presentan los resultados de los experimentos preliminares.

Tabla 44 - Resultados de la ejecución del algoritmo GBHS2 Tagger con diferentes contextos

Contexto	Algoritmo	Núm. de oraciones	Precisión (%)	Desviación estándar
Contexto 4	**GBHS2Pred2 0.5 0.5**	**5000**	**94.2810022**	**7.897156773**
Contexto 5	GBHS2PredSuce 0.5 0.5	5000	94.14608674	7.702875326
Contexto 2	GBHS2PredSuc2 0.5 0.5	5000	94.09230958	7.68880168
Contexto 1	GBHS2Pred2Suc 0.5 0.5	5000	94.0811572	7.722820905
Contexto 3	GBHS2Suc2 0.5 0.5	5000	94.07903069	7.902208265

Como se puede apreciar, los resultados muestran que el algoritmo GBHS2Pred2 (Contexto 4) supera en precisión a los demás etiquetadores, incluso al trigrama presentado en [25] para el caso del castellano, pero, una prueba t de Student muestra que con un α=0,05 y 58 grados de libertad no se encuentran diferencias estadísticas entre las medias obtenidas. Con este resultado orientador (no definitivo), el paso siguiente fue la ejecución del algoritmo en su versión mejorada GBHS2Pred2 con el

dataset completo del corpus IULA, utilizando igualmente validación cruzada de 5 folders, los resultados se muestran en la Tabla 6 de la sección 6, donde se pueden apreciar las diferencias con el algoritmo memético GBHS 2 Tagger propuesto en [25].

6. Resultados de los experimentos realizados

Los experimentos fueron realizados sobre los algoritmos metaheurísticos: 1) Memético GBHS2Pred2 con la mejora del contexto presentada en la sección 5, 2) los metaheurísticos Azar (búsqueda aleatoria), GBHS Tagger, GBHS2 Tagger, GBHS3 Tagger, HSTagger, HSTagger2 y HSTagger3 presentados en [25]. Todos los algoritmos se ejecutaron sobre el dataset IULA de la sección 4.

Para la ejecución de los experimentos se utilizó un modelo cliente – servidor, con una base de datos en la nube (servidor), en donde los clientes (máquinas) solicitan cierto número de tareas, cada tarea tiene la frase y esta es ejecutada 30 veces en la maquina local, una vez finalizada la tarea se hace conexión con la base de datos en la nube y se graban los resultados, así hasta terminar las tareas solicitadas. Este proceso se repite por parte de los clientes hasta que se terminan las tareas por ejecutar en la base datos. Todos los experimentos se ejecutaron utilizando validación cruzada de 5 folders. En la Tabla 4, se muestra los conjuntos de datos (datasets) de prueba y entrenamiento para cada folder. Los parámetros usados para todos los algoritmos de la línea base fueron los definidos en [25].

Tabla 45 - Conjunto de datos de prueba y entrenamiento para los experimentos

Folder Prueba	Oraciones Prueba	Palabras Prueba	Folder Entrenamiento	Palabras Entrenamiento	Palabras comunes	Palabras desconocidas
1	8416	16316	2, 3, 4, 5	65521	12422	3894 (24%)
2	8416	16357	1, 3, 4, 5	65480	12494	3863 (24%)
3	8416	16466	1, 2, 4, 5	65371	12506	3960 (24%)
4	8416	16290	1, 2, 3, 5	65547	12409	3881 (24%)
5	8415	16408	1, 2, 3, 4	65429	12509	3899 (24%)

La Tabla 5 muestra la precisión obtenida en orden ascendente y los valores de la desviación estándar para cada uno de los algoritmos, también se puede apreciar que los mejores resultados son los del algoritmo GBHS Tagger 2 con la mejora propuesta. La medida de precisión usada para la evaluación de los algoritmos es la presentada en la Ecuación 2. **Medida de Precisión.** Fuente: **[24]**.

$$Precisión = \frac{Número\ Palabras\ correctamente\ etiquetadas}{Número\ de\ palabras\ de\ la\ frase}$$

Ecuación 16 - Medida de Precisión. Fuente: [24]

Tabla 46 - Resultados de desempeño de los algoritmos en dataset IULA para Castellano

Algoritmo	Parámetro	Num Oraciones	Precisión (%)	Desviación	Precisión (%) Palabras desconocidas	Desviación (Palabras desconocidas)
Azar	-	42079	94.8647	6.7539	90.0717	7.9378
HSTagger 3	-	42079	95.3244	6.8978	89.2502	7.7518
HSTagger	-	42079	96.9699	5.8722	93.1611	7.5736
HSTagger 2	-	42079	96.982	5.8321	93.1185	7.4774
GBHS Tagger 2	0.0	42079	97.1298	5.8183	93.4895	7.5877
GBHS Tagger 3	0.0	42079	97.158	5.8163	93.5645	7.6159
GBHSTagger	0.0	42079	97.1753	5.8456	93.6521	7.7179
GBHSTagger	0.3	42079	97.4002	5.8272	94.1479	7.8776
GBHSTagger	0.5	42079	97.4002	5.8272	94.1479	7.8776
GBHSTagger	0.7	42079	97.4002	5.8272	94.1479	7.8776
GBHS Tagger 3	0.3	42079	97.4124	5.7942	94.1507	7.8123
GBHS Tagger 3	0.5	42079	97.4124	5.7942	94.1507	7.8123
GBHS Tagger 3	0.7	42079	97.4124	5.7942	94.1507	7.8123
GBHS Tagger 2	0.3	42079	97.4306	5.7844	94.1928	7.8055
GBHS Tagger 2	0.5	42079	97.4306	5.7844	94.1928	7.8055
GBHS Tagger 2	0.7	42079	97.4306	5.7844	94.1928	7.8055
GBHS2Pred2	0.5	42079	**97.6051**	**5.6494**	**94.4238**	**7.8071**

A los resultados de los experimentos realizados se le aplicó la prueba estadística no paramétrica de Friedman, que confirma que el GBHS2Pred2 supera en rendimiento a los demás algoritmos. La Tabla 6, muestra los resultados obtenidos en la prueba. Esta prueba tiene 17 grados de libertad (82.319298) y un valor p de 1.9663E-10, que lo hace estadísticamente significativo.

Tabla 47 - Resultados Test de Friedman. Fuente: Propia Uso KEEL (Alcalá-Fdez *et al*., 2009)

Algoritmo	Ranking Friedman	Algoritmo	Ranking Friedman	Algoritmo	Ranking Friedman
GBHS2Pred2 0.5 0.5	**1**	GBHS3 0.7 0.5	6.6	GBHS2 0.0 0.5	13
GBHS2 0.3 0.5	3	GBHS 0.3	8.4	HSTagger2 0.5	14.2
GBHS2 0.5 0.5	3	GBHS 0.5	8.4	HSTagger	14.8
GBHS2 0.7 0.5	3	GBHS 0.7	8.4	HSTagger3 0.5	16.2
GBHS3 0.3 0.5	6.6	GBHS 0.0	11	Azar	16.8
GBHS3 0.5 0.5	6.6	GBHS3 0.0 0.5	12		

7. Conclusiones

Los contextos (ventanas) son indispensables al momento de realizar los experimentos, ya que al aportar más información al etiquetado se mejora la precisión, sin embargo, esto depende de las características propias de la lengua y el comportamiento del algoritmo sobre estos contextos. El algoritmo memético GBHS2 Tagger demostró que, al incluir más conocimiento sobre el problema de etiquetado como información contextual, se obtienen resultados superiores sobre los otros algoritmos presentados en este trabajo; sin embargo, también se aprecia que el desempeño de los otros algoritmos metaheurísticos es muy cercano al mejor. El uso de validación cruzada de 5 folders favoreció la independencia entre los datos de entrenamiento y de prueba.

Como trabajo futuro, se propone primero, la adaptación de otros algoritmos metaheurísticos al problema de etiquetado, como Jaya [27], Particle Swarm Optimization (PSO) [30], Hill Climbing [32], Hill Climbing with Random Restart [33] y Evolución Diferencial [89]; y segundo, evaluar los algoritmos metaheurísticos sobre otros corpus como el Brown [20] y el Nasa Yuwe [26], con el fin de observar el desempeño sobre las ventanas contextuales del algoritmo mejorado en este trabajo.

8. Conclusiones

Los autores agradecen a la Universidad del Cauca, por toda su colaboración.

9. Referencias

[1] G. G. Chowdhury, “Natural language processing,” *Annu. Rev. Inf. Sci. Technol.*, vol. 37, no. 1, pp. 51–89, Jan. 2005, doi: 10.1002/aris.1440370103.

[2] A. Ekbal and S. Saha, “Simulated annealing based classifier ensemble techniques: Application to part of speech tagging,” *Inf. Fusion*, vol. 14, no. 3, pp. 288–300, 2013, doi: 10.1016/j.inffus.2012.06.002.

[3] K. Ravi and V. Ravi, “A survey on opinion mining and sentiment analysis: Tasks, approaches and applications,” *Knowledge-Based Syst.*, vol. 89, no. C, pp. 14–46, 2015, doi: 10.1016/j.knosys.2015.06.015.

[4] M. Bordoloi and S. K. Biswas, “Graph-Based Sentiment Analysis Model for E-Commerce Websites’ Data,” *Cogn. Informatics Soft Comput. Proceeding CISC 2017*, pp. 453–462, 2019, doi: 10.1007/978-981-13-0617-4.

[5] J. Ma, H. Liu, D. Huang, and W. Sheng, “An English part-of-speech tagger for machine translation in business domain,” *7th Int. Conf. Nat. Lang. Process. Knowl. Eng.*, pp. 183–189, 2011, doi: 10.1109/NLPKE.2011.6138191.

[6] S. Huet, G. Gravier, and P. Sébillot, “Morphosyntactic resources for automatic

speech recognition," *6th Int. Conf. Lang. Resour. Eval. Lr. 2008*, pp. 692–698, 2008.

[7] R. Karimpour *et al.*, "Improving persian information retrieval systems using stemming and part of speech tagging," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5706, pp. 89–96, 2009, doi: 10.1007/978-3-642-04447-2_10.

[8] T. Güngör, *Handbook of Natural Language Processing (second edition)*. 2011.

[9] D. Jurafsky and J. H. Martin, "Speech and Language Processing," *Speech Lang. Process. An Introd. to Nat. Lang. Process. Comput. Linguist. Speech Recognit.*, vol. 21, pp. 151–176, 2009, doi: 10.1162/089120100750105975.

[10] E. Brill, "A simple rule-based part of speech tagger," *Proc. third Conf. Appl. Nat. Lang. Process.*, 1992, doi: 10.3115/974499.974526.

[11] D. Q. Nguyen, D. Q. Nguyen, D. D. Pham, and S. B. Pham, "A robust transformation-based learning approach using ripple down rules for part-of-speech tagging," *AI Commun.*, vol. 29, no. 3, pp. 409–422, 2016, doi: 10.3233/AIC-150698.

[12] L. Araujo, "Symbiosis of Evolutionary Techniques and Statistical Natural Language Processing," *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 14–27, 2004, doi: 10.1109/TEVC.2003.818189.

[13] A. P. Silva, A. Silva, and I. Rodrigues, *Part-of-speech tagging using evolutionary computation*, vol. 512.Spring. 2014.

[14] A. Paul, B. S. Purkayastha, and S. Sarkar, "Hidden Markov Model based Part of Speech Tagging for Nepali language," *undefined*, 2015.

[15] A. Ratnaparkhi, "A Maximum Entropy Model for Part-Of-Speech Tagging," *Proc. Conf. Empir. Methods Nat. Lang. Process. EMNLP-96*, pp. 133–142, 1996, doi: 10.1007/s00280-013-2178-x.

[16] T. Brants, "A Statistical Part-of-Speech Tagger," vol. 5, pp. 1–7, 2000.

[17] D. Surendran and G.-A. Levow, "Dialog Act Tagging with Support Vector Machines and Hidden Markov Models," *Interspeech 2006 9th Int. Conf. Spok. Lang. Process.*, pp. 1950–1953, 2006.

[18] H. Schmid, "Part-of-speech tagging with Neural Networks," *Eur. J. Cancer Prev.*, vol. 27, no. 4, pp. 296–302, 1994, doi: 10.1097/CEJ.0000000000000354.

[19] L. Araujo, G. Luque, and E. Alba, "Metaheuristics for Natural Language Tagging," pp. 889–900, 2004, doi: 10.1007/978-3-540-24854-5_90.

[20] W. N. Francis and H. Kucera, "Brown Corpus Manual," 1979. [Online].

Available: http://clu.uni.no/icame/manuals/BROWN/INDEX.HTM#bc8. [Accessed: 03-Dec-2018].

[21] A. P. Silva, A. Silva, and I. Rodrigues, “BioPOS : Biologically Inspired Algorithms for POS Tagging,” *Proc. First Int. Work. Optim. Tech. Hum. Lang. Technol.*, vol. 2, no. December, pp. 1–16, 2013.

[22] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, “Building a large annotated corpus of English: the penn treebank,” *Comput. Linguist.*, vol. 19, no. 2, pp. 313--330, 1993, doi: 10.1162/coli.2010.36.1.36100.

[23] “Mac-Morpho.” [Online]. Available: http://nilc.icmc.usp.br/macmorpho/. [Accessed: 05-Dec-2018].

[24] R. Forsati and M. Shamsfard, “Novel harmony search-based algorithms for part-of-speech tagging,” *Knowl. Inf. Syst.*, vol. 42, no. 3, pp. 709–736, 2014, doi: 10.1007/s10115-013-0719-6.

[25] L. M. Sierra Martínez, C. A. Cobos, and J. C. Corrales, “Memetic algorithm based on global-best harmony search and hill climbing for part of speech tagging,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10682 LNAI, pp. 198–211, 2017, doi: 10.1007/978-3-319-71928-3_20.

[26] L. M. Sierra Martínez, C. A. Cobos, C. J. Muñoz Corrales, T. Curieux Rojas, E. Herrera-viedma, and D. H. Peluffo-ordóñez, “Building a Nasa Yuwe Language Corpus and Tagging with a Metaheuristic Approach,” *19th Int. Conf. Intell. Text Process. Comput. Linguist.*, vol. 22, no. 3, pp. 881–894, 2018, doi: 10.13053/CyS-22-3-3018.

[27] P. Singh and H. Chaudhary, “A Modified Jaya Algorithm for Mixed-Variable Optimization Problems,” *J. Intell. Syst.*, vol. 0, no. 0, pp. 1–21, 2018, doi: 10.1515/jisys-2018-0273.

[28] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, “Metaheuristic research: a comprehensive survey,” *Artif. Intell. Rev.*, no. January, pp. 1–43, 2018, doi: 10.1007/s10462-017-9605-z.

[29] W. G. M. Wolpert, David H., “No Free Lunch Theorems for Optimization,” *Encycl. GIS*, vol. 1, no. 1, pp. 240–240, 2008, doi: 10.1109/4235.585893.

[30] F. Neri, E. Mininno, and G. Iacca, “Compact particle swarm optimization,” *Inf. Sci. (Ny).*, vol. 239, pp. 96–121, 2013, doi: 10.1016/j.ins.2013.03.026.

[31] T. Rojas, “Desde arriba y por abajo construyendo el alfabeto nasa. La experiencia de la unificación del alfabeto de la lengua páez (nasa yuwe) en el Departamento del Cauca Colombia.,” *Construyendo El Alf.*, vol. 1, no. 1, p. 15, 2001.

[32] J. Brownlee, *Clever Algorithms*. 2011.

[33] S. Luke, *Essentials of Metaheuristics*. 2015.

[34] K. Schwaber and J. Sutherland, “La Guía de Scrum. La Guía Definitiva de Scrum: Las Reglas del Juego.,” p. 22, 2017.

[35] K. S. Pratt, “Design Patterns for Research Methods: Iterative Field Research,” *AAAI Spring Symp. Exp. Des. Real*, no. 1994, pp. 1–7, 2009.

[36] I. Project Management Institute, *Institute, A guide to the Project Management Body of Knowledge, 5 ed., Project Management Institute*. 2017.

[37] D. Cutting, J. Kupiec, J. Pedersen, and S. Penelope, “A Practical Part-of-Speech Tagger,” *Proc. Conf.*, pp. 133–140, 1992.

[38] R. Forsati, “Cooperation of Evolutionary and Statistical,” no. Aisp, pp. 550–555, 2012.

[39] I. Boussaïd, J. Lepagnot, and P. Siarry, “A survey on optimization metaheuristics,” *Inf. Sci. (Ny).*, vol. 237, pp. 82–117, Jul. 2013, doi: 10.1016/j.ins.2013.02.041.

[40] C. Cotta, “Una Visión General de los Algoritmos Meméticos.”

[41] S. Luke, *Essentials of Metaheuristics*. 2015.

[42] E. R. R. Kato, G. D. de A. Aranha, and R. H. Tsunaki, “A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing,” *Comput. Ind. Eng.*, vol. 125, pp. 178–189, Nov. 2018, doi: 10.1016/j.cie.2018.08.022.

[43] Y. Zhang, S. Wang, and G. Ji, “A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications,” *Mathematical Problems in Engineering*, vol. 2015. Hindawi Limited, 2015, doi: 10.1155/2015/931256.

[44] E. R. R. Kato, G. D. de A. Aranha, and R. H. Tsunaki, “A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing,” *Comput. Ind. Eng.*, vol. 125, pp. 178–189, Nov. 2018, doi: 10.1016/j.cie.2018.08.022.

[45] R. Venkata Rao, “Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems,” *Int. J. Ind. Eng. Comput.*, vol. 7, no. 1, pp. 19–34, Dec. 2016, doi: 10.5267/j.ijiec.2015.8.004.

[46] R. Kennedy, J., & Eberhart, “Particle swarm optimization,” *Proc. Proc. IEEE Int. Conf. neural networks*, vol. Vol. 4, pp. 1942–1948, 1995.

[47] M. Sevkli and A. R. Guner, “A Continuos Particle Swarm Optimization Algorithm for Uncapacited Facility Location Problem,” pp. 316–323, 2006.

[48] Q. Pan, M. F. Tasgetiren, and Y. Liang, “A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem,” vol. 35, pp. 2807–2839, 2008, doi: 10.1016/j.cor.2006.12.030.

[49] C. J. Liao, Chao-Tang Tseng, and P. Luarn, “A discrete version of particle swarm optimization for flowshop scheduling problems,” *Comput. Oper. Res.*, vol. 34, no. 10, pp. 3099–3111, Oct. 2007, doi: 10.1016/j.cor.2005.11.017.

[50] M. Clerc, “Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem,” Springer, Berlin, Heidelberg, 2004, pp. 219–239.

[51] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, “A novel particle swarm optimization algorithm with adaptive inertia weight,” *Appl. Soft Comput. J.*, vol. 11, no. 4, pp. 3658–3670, 2011, doi: 10.1016/j.asoc.2011.01.037.

[52] A. Abunaser, I. A. Doush, N. Mansour, and S. Alshattnawi, “Underwater image enhancement using particle swarm optimization,” *J. Intell. Syst.*, vol. 24, no. 1, pp. 99–115, Mar. 2015, doi: 10.1515/jisys-2014-0012.

[53] M. G. H. Omran and M. Mahdavi, “Global-best harmony search,” *Appl. Math. Comput.*, vol. 198, no. 2, pp. 643–656, 2008, doi: 10.1016/j.amc.2007.09.004.

[54] A. Alhasan and A. T. Al-taani, “POS Tagging for Arabic Text Using Bee Colony Algorithm,” *Procedia Comput. Sci.*, pp. 158–165, 2018, doi: 10.1016/j.procs.2018.10.471.

[55] M. El-Haj and R. Koulali, “KALIMAT a multipurpose Arabic Corpus,” *Second Work. Arab. Corpus Linguist.*, 2013.

[56] A. Al-Taani and S. A. Al-Rub, “A rule-based approach for tagging non-vocalized Arabic words,” *Int. Arab J. Inf. Technol.*, vol. 6, no. 3, pp. 320–328, 2009.

[57] S. Petrov, D. Das, and R. McDonald, “A Universal Part-of-Speech Tagset,” 2011, doi: 10.1038/hdy.2008.34.

[58] R. Forsati, M. Shamsfard, and P. Mojtahedpour, “An efficient meta heuristic algorithm for POS-tagging,” *Proc. - 5th Int. Multi-Conference Comput. Glob. Inf. Technol. ICCGI 2010*, pp. 93–98, 2010, doi: 10.1109/ICCGI.2010.42.

[59] E. Alba, G. Luque, and L. Araujo, “Natural language tagging with genetic algorithms,” *Inf. Process. Lett.*, vol. 100, no. 5, pp. 173–182, Dec. 2006, doi: 10.1016/J.IPL.2006.07.002.

[60] R. Forsati and M. Shamsfard, “Hybrid PoS-tagging: A cooperation of evolutionary and statistical approaches,” *Appl. Math. Model.*, vol. 38, no. 13, pp. 3193–3211, Jul. 2014, doi: 10.1016/J.APM.2013.11.047.

[61] A. P. Silva, A. Silva, and I. Rodrigues, “An Approach to the POS Tagging Problem Using Genetic Algorithms,” in *In Computation Intellingence*, 2012, pp.

3–17.

[62] A. P. Silva, A. Silva, and I. Rodrigues, “A New Approach to the POS Tagging Problem Using Evolutionary Computation,” *Proc. Recent Adv. Nat. Lang. Process.*, no. September, pp. 619–625, 2013.

[63] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, “A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA,” *IEEE Trans. Evol. Comput.*, vol. 12, no. 3, pp. 269–283, Jun. 2008, doi: 10.1109/TEVC.2007.900837.

[64] T. Curieux Rojas, “Esbozo gramatical de la lengua nasa (lengua páez),” *El Leng. en Colomb.*, vol. 1: Realida, no. Ed., Bogotá, Academia Colombiana de la Lengua e Instituto Caro y Cuervo, 2012.

[65] H. M. Alonso and D. Zeman, “Universal dependencies for the AnCora treebanks,” *Proces. Leng. Nat.*, vol. 57, pp. 91–98, 2016.

[66] Universal Dependencies, “Universal Dependencies.” [Online]. Available: http://universaldependencies.org/. [Accessed: 17-Dec-2018].

[67] J. Lavid, J. Arús, B. DeClerck, and V. Hoste, “Creation of a High-quality, Register-diversified Parallel (English-Spanish) Corpus for Linguistic and Computational Investigations,” *Procedia - Soc. Behav. Sci.*, vol. 198, pp. 249–256, Jul. 2015, doi: 10.1016/J.SBSPRO.2015.07.443.

[68] K. Toutanova, D. Klein, C. Manning, and Y. Singer, “Feautre-Rich Part-of-Speech Tagging with a Cyclic Dependency Network,” *Proc. 2003 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol.*, vol. 1, no. 4, pp. 173–180, 2003, doi: 10.1007/BF02379273.

[69] U. P. F. Institut Universitari de Lingüística Aplicada (IULA), “IULA Spanish LSP Treebank,” 2012.

[70] EAGLES, “ETIQUETAS EAGLES.” [Online]. Available: http://blade10.cs.upc.edu/freeling-old/doc/tagsets/tagset-es.html. [Accessed: 03-Dec-2018].

[71] I. Zeroual, A. Lakhouaja, and R. Belahbib, “Towards a standard Part of Speech tagset for the Arabic language,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 29, no. 2, pp. 171–178, 2017, doi: 10.1016/j.jksuci.2017.01.006.

[72] H. Schmid, “TreeTagger - a language independent part-of-speech tagger,” 1994. [Online]. Available: http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/treetagger.en.html. [Accessed: 12-Dec-2018].

[73] B. Kabashi and T. Proisl, “A Proposal for a Part-of-Speech Tagset for the Albanian Language,” *Proc. Tenth Int. Conf. Lang. Resour. Eval. (LREC 2016)*,

pp. 4305–4310, 2016.

[74] M.-C. De Marneffe, B. MacCartney, and C. D. Manning, “Generating Typed Dependency Parses from Phrase Structure Parses,” *Proc. Lr.*, 2006, doi: 10.1145/1391729.1391731.

[75] D. Zeman, “Reusable Tagset Conversion Using Tagset Drivers,” *Proc. Lr.*, 2008.

[76] J. Lafferty, A. Mccallum, and F. Pereira, “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data Part of the Numerical Analysis and Scientific Computing Commons Recommended Citation "Conditional Random Fields: Probabilistic Models for Segmenting and Labelin,” *Proc. ICML*, vol. 2001, no. June, pp. 282–289, 2001.

[77] J. Tobar and M. Solano, “Dataset IULA,” 2020. .

[78] “slavpetrov/universal-pos-tags: Automatically exported from code.google.com/p/universal-pos-tags.” [Online]. Available: https://github.com/slavpetrov/universal-pos-tags. [Accessed: 27-Jun-2019].

[79] J. Alcalá-Fdez *et al.*, “KEEL: A software tool to assess evolutionary algorithms for data mining problems,” *Soft Comput.*, vol. 13, no. 3, pp. 307–318, 2009, doi: 10.1007/s00500-008-0323-y.

[80] S. Callander, “Searching and learning by trial and error,” *Am. Econ. Rev.*, vol. 101, no. 6, pp. 2277–2308, 2011, doi: 10.1257/aer.101.6.2277.

[81] A. K. Mishra and D. Shrivastava, “A discrete Jaya algorithm for permutation flow-shop scheduling problem,” vol. 11, pp. 1–14, 2020, doi: 10.5267/j.ijiec.2019.12.001.

[82] S. O. Degertekin, L. Lamberti, and I. B. Ugur, “Sizing, layout and topology design optimization of truss structures using the Jaya algorithm,” *Appl. Soft Comput. J.*, vol. 70, pp. 903–928, Sep. 2018, doi: 10.1016/j.asoc.2017.10.001.

[83] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan, “Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm,” *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1944–1955, 2019, doi: 10.1109/TCYB.2018.2817240.

[84] M. A. Al-Betar, “β-Hill climbing: an exploratory local search,” *Neural Comput. Appl.*, vol. 28, no. 1, pp. 153–168, Dec. 2017, doi: 10.1007/s00521-016-2328-2.

[85] C. Laguna, “Inferencia Paramétrica: Relación entre dos variables cualitativasy cuantitativas.,” *Inst. Aragon. ciencias la salud*, vol. 8, no. 2, p. 2,10, 2016.

[86] M. Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. 2011.

[87] “Insomnia REST Client.” [Online]. Available: https://insomnia.rest/. [Accessed: 31-Mar-2020].

[88] J. Walke, “React – Una biblioteca de JavaScript para construir interfaces de usuario.” [Online]. Available: https://es.reactjs.org/. [Accessed: 05-Apr-2020].

[89] L. M. Sierra, C. Cobos, and J. C. Corrales, “Continuous Optimization Based on a Hybridization of Differential Evolution with K-means,” *Bazzan A., Pichara K. Adv. Artif. Intell.*, vol. 8864, pp. 381–392, Nov. 2014, doi: 10.1007/978-3-319-12027-0_31.

[90] J. J. Tobar, M. A. Solano, L. M. Sierra, and C. A. Cobos, “Etiquetado de partes del discurso sobre un corpus en castellano basado en metaheurísticas,” *Rev. Ibérica Sist. e Tecnol. Inf.*, p. 15, 2020.

[91] D. Moussallem, M. Wauer, and A. N. Ngomo, “Web Semantics : Science , Services and Agents on the World Wide Web Machine Translation using Semantic Web Technologies : A Survey,” *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 51, pp. 1–19, 2018, doi: 10.1016/j.websem.2018.07.001.

[92] C. Garcia-Martinez, F. J. Rodriguez, and M. Lozano, “Ánalisis Empirico del No Free Lunch en Problemas Binarios Reales.” .

Anexo 2

Articulo Final Resultados

2.1 Contenido del articulo

Algoritmo memético para la identificación de partes del discurso utilizando un enfoque metaheurístico.

José Julio Tobar C, Miguel Alexis Solano J, Luz Marina Sierra-M, Carlos Alberto Cobos L.

Resumen: El etiquetado de partes del discurso es una de las tareas más importantes en el preprocesamiento del lenguaje natural y se usa entre otros para el análisis de sentimientos, traducción de texto, reconocimiento de voz y recuperación de información. Esta tarea se enfrenta a tres retos principales relacionados con la ambigüedad de las palabras, el tamaño del conjunto de etiquetas y el etiquetado de palabras desconocidas. Este articulo presenta la propuesta de un algoritmo memético basado en GBHS como optimizador global y HC como optimizador local, también presenta la construcción de un dataset en castellano y la comparación de varios algoritmos metaheurísticos del estado del arte sobre los corpus IULA para el castellano, Brown para el Inglés y Nasa Yuwe para el Nasa, en donde el memético propuesto obtiene un mejor desempeño.

Palabras-clave: Global best harmony search; Algoritmos metaheurísticos; Algoritmos meméticos; Identificación de partes del discurso; Corpus etiquetado IULA.

Abstract: The Part of Speech Tagging is one of the most important tasks in the natural language preprocessing and is used among others for sentiment analysis, text translation, voice recognition and information retrieval. This task faces three main challenges related to the ambiguity of words, the size of the Tagset and the labeling of unknown words. This article presents the proposal of a memetic algorithm based on GBHS as global optimizer and HC as local optimizer, it also presents the construction of a dataset in Spanish and the comparison of several state-of-art metaheuristic algorithms on IULA corpus for Spanish, Brown for English and Nasa Yuwe for Nasa, where the proposed memetic performs better.

Keywords*:* Metaheuristic algorithms; Memetic algorithm; Corpus tagged IULA; Part of Speech Tagging; Global-best Harmony Search; Hill-Climbing

1. Introducción

El etiquetado de partes del discurso(Part-of-Speech tagging, POST) es un área del procesamiento de lenguaje natural (PLN) que se encarga de asignar a cada palabra de una oración sus posibles categorías léxicas, de acuerdo a su contexto [8], y se constituye en una de las tareas más importantes en el pre-procesamiento de la mayoría de las aplicaciones de PLN. El etiquetado se convierte en una tarea compleja, al enfrentar principalmente tres grandes retos [8], [9] a saber: 1) La ambigüedad de palabras. 2) El tamaño del conjunto de etiquetas. 3) El etiquetado de palabras desconocidas.

En los últimos años el problema de etiquetado ha sido abordado desde la perspectiva de los algoritmos metaheurísticos, tanto desde una perspectiva estadística como basada en reglas. Estas propuestas metaheurísticas han obtenido un desempeño sobresaliente en comparación con los enfoques descritos anteriormente, como se puede apreciar en: Araujo *et al*. [19] que utilizan un algoritmo genético, el recocido simulado y el algoritmo CHC, para el problema de etiquetado; Silva *et al*.[21], que abordan el etiquetado como un problema de optimización usando el algoritmo de optimización por enjambre de partículas (Particle Swarm Optimization, PSO). Forsati *et al*. [24] que utilizan la metaheurística de la Búsqueda Armónica (Harmony Search, HS), abordando el problema de etiquetado desde un enfoque estadístico; entre otros trabajos. Los enfoques metaheurísticos se encuentran aplicados a corpus etiquetados de lenguas como el inglés y otras lenguas no tradicionales como: el árabe, Bengalí (Bangladés), hindi (India), Telugu (India) y Nasa Yuwe (lengua indígena de Colombia).

En esta investigación se aborda el problema de etiquetado desde la perspectiva de los algoritmos metaheurísticos, con el objetivo de proponer mejoras al memético

propuesto en [25] y adaptar otras metaheurísticas al problema de etiquetado, configurando sus evaluaciones, sobre los corpus etiquetado IULA del castellano [69], corpus Brown de la lengua inglés [20] y Nasa Yuwe [26].

Adicionalmente en este trabajo, se presenta el diseño e implementación de un prototipo software para etiquetar oraciones de los tres corpus con los algoritmos seleccionados.

El resto del articulo está organizado de la siguiente manera: Sección 2, trabajos más relevantes que abordan el problema de etiquetado desde una perspectiva metaheurística, la descripción del conjunto de etiquetas y los corpus utilizados; Sección 3, proceso metodológico seguido; Sección 4, se presenta los algoritmos adaptados; Sección 5 las propuestas para mejorar el algoritmo memético; Sección 6, experimentos realizados sobre los algoritmos metaheurísticos con sus respectivos resultados; Sección 7 presenta el prototipo software de partes del discurso y finalmente, la sección 8, presenta las conclusiones y algunas ideas de trabajos futuros.

2. Contexto

2.1. Algoritmos metaheurísticos y meméticos

El algoritmo PSO fue propuesto por Kennedy y Eberhart [46], es una de las técnicas de optimización más utilizadas que ha obtenido muy buenos resultados de una manera más rápida y económica que otros métodos [43]. PSO ha sido aplicado con éxito a una amplia gama de aplicaciones, como la optimización de funciones, la asignación de tareas y los problemas de programación [47]; el algoritmo original de [46] está diseñado para resolver problemas de optimización continuo, esta ha sido una de las cuestiones para el diseño del algoritmo a problemas discretos, debido a que las ecuaciones del PSO original no se pueden usar para generar una solución discreta, ya que las posiciones que se generan tienen un valor real [48]; sin embargo, existen propuestas que han diseñado el algoritmo PSO discreto dependiendo del problema a optimizar, algunos son: Pan *et al* en [48], Boussaid *et al* en [49] y Crec *et al* en [50]. El algoritmo PSO, es una metaheurística poblacional motivada por el comportamiento colectivo inteligente de algunos animales. Cada solución potencial se denomina partícula, el conjunto de partículas es conocido como enjambre, donde la posición de cada partícula cambia en función de su propia experiencia (pBest) y la experiencia del enjambre (gBest) [51]; en cada iteración la velocidad y la posición son actualizados.

El algoritmo Jaya [45], es presentado como un algoritmo simple, que únicamente requiere parámetros de control comunes; es considerado muy fácil de usar en

contraste con otros algoritmos evolutivos, debido a que este algoritmo encuentra soluciones más óptimas, además de no requerir parámetros de control específicos [28]. En su naturaleza tiene como objetivo principal acercarse al éxito, es decir, encontrar la más óptima de las soluciones en el menor tiempo, pero también a su vez siempre está intentando alejarse del fracaso, es decir, de la peor de las soluciones, generando un balance óptimo entre exploración y explotación. En el estado del arte no se evidencia trabajos relacionados con la aplicación de Jaya al problema de etiquetado. El algoritmo Jaya en su versión original, ha sido desarrollado para resolver problemas de optimización con variables de diseño continuo [27], pero recientemente este algoritmo se ha implementado para resolver problemas de optimización de variables discretas y enteras [82]. Una estrategia de discretización es la presentada en [83], en donde, se consideran tres números aleatorios binarios (es decir, 0 a 1) considerados para el propósito de selección. En la **Ecuación 17,** se presenta la función de actualización del algoritmo discreto DJaya.

$$X'_i(t+1) = b_1 X_i(\mathrm{t}) + b_2 X_B(\mathrm{t}) + b_3 X_W(\mathrm{t}), \qquad i = 1, 2, \dots, N_p$$

$$with\ b_1, b_2, b_3 \in \{0,1\} \text{ and } \sum_{j=1}^{3} b_j = 1$$

Ecuación 17. Discretización para el algoritmo Jaya**. Fuente:** [83].

En la **Ecuación 17,** se puede ver que existe la posibilidad de que se seleccione uno y solo un elemento, dependiendo de los valores de los números aleatorios b1, b2 y b3 para una nueva solución, al mismo tiempo cuando una operación es seleccionada desde X_i, X_B o X_W debe verificarse que esta operación no se haya seleccionado, para asegurarse de que no se repita una solución.

2.2. Trabajos relacionados

Una vez realizada la revisión de la literatura en las diferentes bases de datos y motores de búsqueda como: IEEE, Springer, ACM, Scopus, ScienceDirect, Google Académico y ProQuest, se encontraron los siguientes trabajos relevantes:

En 2018, Alhasan y Al-taani [54], presenta un etiquetador, utilizando el algoritmo de optimización por colonia de abejas (BCO) para la lengua árabe; en 2017, Sierra et al. [25] presentan un etiquetador basado en Global-Best Harmony Search (GBHS) que incluye conocimiento del problema mediante una estrategia de optimización local basada en el algoritmo Hill Climbing [32], evaluado sobre una lengua tradicional como el Inglés y una no tradicional como el Nasa Yuwe; en 2015, Forsati y Shamsfard [24], presentaron dos mejoras del etiquetador HSTAGger [60], que incrementan la eficiencia de búsqueda y mejoran la selección de nuevas soluciones para la memoria armónica,

fue evaluado sobre el corpus Brown [20] y PennTreenbank [22]; en 2013, Silva et al. [62] dividen el problema de etiquetado en dos tareas: una de aprendizaje y otra optimización, que ayuda al descubrimiento de reglas usando la computación evolutiva, utilizando dos algoritmos, un algoritmo genético (GA-Tagger) y un PSO binario (PSO-Tagger), que fue evaluado sobre el corpus Brown.

En 2018, [26], construyeron el corpus etiquetado para el nasa Yuwe, el cual contiene 1176 palabras etiquetadas, consolidando un total de 175 oraciones etiquetadas, este corpus utiliza el conjunto de etiquetas universal [57]; En 2012, el Institut Universitari de Lingüística Aplicada (IULA) [69] presentó el corpus IULA Spanish LSP Treebank que consta de 40000 oraciones de la lengua castellana, tomadas del IULA technical corpus. Las etiquetas se basan en el conjunto de etiquetas ***EAGLES*** [70] para el español, el cual contiene 577 etiquetas. Este corpus en 2020, [90] fue estructurado como un dataset de 42079 oraciones y mapeado al conjunto de etiquetas Petrov [57], además fue estructurado en una base de datos, listo para realizar el etiquetado de partes del discurso; en 1979, Francis y Kucera [20] presentaron el corpus Brown, que es una recopilación de 500 archivos de texto plano en Inglés. El conjunto de etiquetas está conformado por 87 categorías, las cuales se subdividen, obteniendo un total de 472 etiquetas.

En 2012, Petrov *et al.* [57], presentan un conjunto de 12 etiquetas comunes para 25 lenguajes (a partir de 25 conjuntos de etiquetas) provenientes de trabajos previos buscando mejorar la precisión de los etiquetadores a través de varios lenguajes. Esta propuesta ha sido muy aceptada y utilizada en diferentes trabajos de identificación de partes del discurso [76].

Después de hacer la revisión del estado del arte se pudo encontrar que: 1) el problema de etiquetado está siendo cada vez más explorado desde diferentes enfoques. 2) El uso de algoritmos metaheurísticos en este problema, ha ido tomando fuerza en los últimos años (2010-2020). 3) El idioma inglés y el conjunto de etiquetas Universal [57] son los más utilizados para la evaluación de desempeño de los etiquetadores.

3. Metodología

Para esta investigación se realizaron tres ciclos: Para el primer y segundo ciclo, se utilizó el Patrón Iterativo de Investigación (Iterative Research Pattern, IRP) propuesto por Pratt [35] y para el tercer ciclo se utilizó SCRUM [34]. El primer ciclo, se enfocó en la adaptación de algoritmos metaheurísticos al problema de etiquetado y la selección del mejor; el segundo ciclo, en la propuesta de una nueva versión de un algoritmo

memético; y, el tercer ciclo, en el diseño y desarrollo de un prototipo software para el etiquetado de oraciones de los tres corpus con los algoritmos adaptados.

Tabla 48. Formato de término del corpus IULA. **Fuente:** Elaboración propia.

	Observación	**Identificación**	**Desarrollo**	**Pruebas**
Ciclo 1	*Revisión de metaheurísticas e investigación de corpus en castellano.*	*Selección del corpus y mapeo de etiquetas al etiquetado universal.*	*Diseño e implementación de la base de datos para el corpus IULA. Configuración y ejecución de experimentos sobre dataset IULA*	*Análisis y discusión de resultados de precisión obtenidos.*
Ciclo 2	*Exploración de ventanas contextuales para el castellano.*	*Definición de los contextos implementados.*	*Implementación de los contextos sobre el mejor algoritmo del ciclo 1. Ejecución de los experimentos y selección de la mejor versión preliminar del algoritmo.*	*Análisis de resultados y emisión de conclusiones*
	Product Backlog	**Sprint 1**	**Sprint 2**	**Despliegue**
Ciclo 3	*Desarrollar un api Web y una aplicación web adaptativa*	*Desarrollo del servicio web en ASP.NET*	*Desarrollo de la aplicación web en react.js*	*Publicación del servicio web en Azure virtual machine Publicación de la aplicación web in AWS*

4. Adaptación de algoritmos metaheurísticos al problema de etiquetado

4.1. Integración de corpus de etiquetado

Una vez realizada la construcción del dataset del corpus IULA, fue requerida la integración de los tres corpus en una base de datos de corpus. En la **Figura 21**, se presenta el modelo conceptual de la Base de Datos de Corpus integrados.

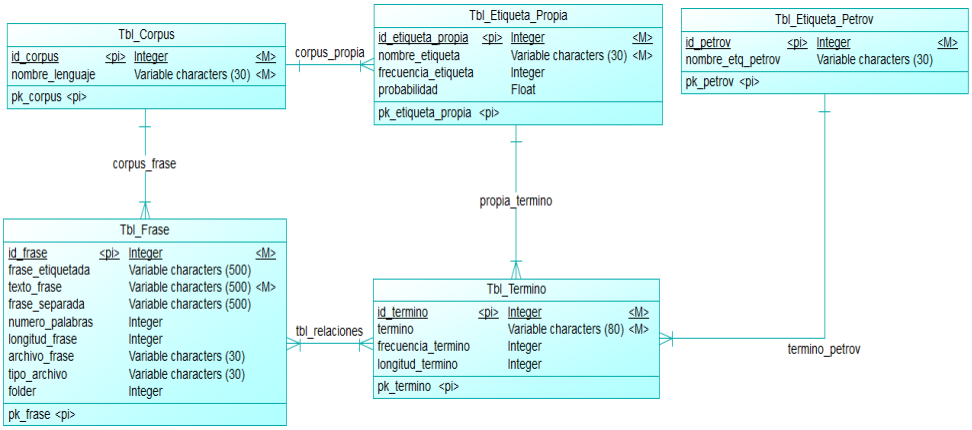


Figura 21. Modelo Conceptual de la Base de Datos de Corpus Integrados

Partiendo del modelo conceptual presentado en la figura 1, se generó el modelo físico de la base de datos para el motor SQL Server 2017, en [77], se encuentran el backup de la base de datos que incluyen estructura y datos. Un aspecto a resaltar en este modelo es la tabla de relaciones (tbl_relaciones) donde se encuentra la información de cada frase de los tres corpus, relacionada con los términos y el orden de cada termino en la frase, además el contexto a tener en cuenta para el término que se va a etiquetar, es decir, las etiquetas de sus predecesores, sucesores según la ventana a tener en cuenta, cada relación de la tabla relaciones, además guarda la llave foránea con su corpus correspondiente (id_corpus).

Como resultado del procesamiento de los tres corpus, se logró consolidar una base de datos con 95355 frases y 139755 términos, es decir, que la tabla de relaciones tiene un total de 1753100 registros. Dada la cantidad de registros y la cantidad de consultas que debe atender, se llevó a cabo un proceso de indexación sobre las tablas, para las llaves con mayor número de relaciones.

4.2. Descripción de la solución utilizada

La solución utilizada para la adaptación de los algoritmos metaheurísticos y meméticos fue la utilizada en [25], que es un trigrama, es decir, al momento de asignar la etiqueta de la palabra se tenía en cuenta la etiqueta de la palabra predecesora y la palabra sucesora, como se muestra en la **Figura 22.**

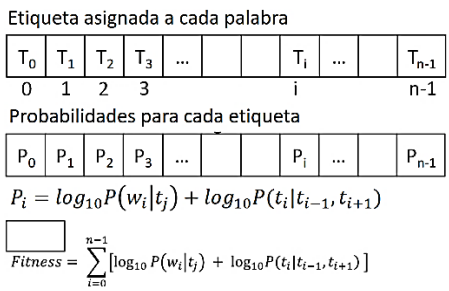


Figura 22. Representación de la solución.

4.3 Adaptación del algoritmo Particle Swarm Optimization PSO

En la revisión de estado del arte se pudo apreciar que el algoritmo PSO se ha utilizado en el problema de etiquetado desde un enfoque basado en reglas [62], donde obtuvo un buen desempeño, pero no ha sido utilizado desde el enfoque estadístico, por tanto, fue seleccionado en esta investigación para su adaptación al problema de etiquetado.

La adaptación del algoritmo PSO al problema de etiquetado se realizó de la siguiente manera: 1) se implementó el algoritmo PSO para problemas discretos, teniendo en cuenta los parámetros originales $W, C1, C2$ y el parámetro de preservación de solución P; 2) la partícula se representa como en la **Figura 22.**

A continuación, en el **Algoritmo 1,** se presenta el pseudocódigo de PSOTagger adaptado para el problema de etiquetado.

Algoritmo 1. PSOTagger

1. Definición de parámetros: $TamPoblacion$, MaxGeneraciones, W, C1, C2, P
2. Inicialización aleatoria de la $Población$
3. Definición de la mejor partícula global (G_{best})
4. Definición de P_{best} para cada partícula en el enjambre
5.
6. **para** $i = 1$ to MaxGeneraciones **hacer**
7. **para** $j = 1\ to\ TamPoblacion$ **hacer** /* para cada partícula */
8. Evaluar la función fitness de la partícula (P_j)
9. **si** (fitness (P_j) > fitness (P_{best}))
10. $P_{best} \leftarrow P_j$
11. **fin_si**
12. **fin_para**
13. Definir $G'_{best} \leftarrow Poblacion^{best}$ /* se define el mejor de la población actual*/
14. **si** (fitness (G'_{best}) > fitness (G_{best})) /* mejor actual es mayor a mejor anterior*/
15. $G_{best} \leftarrow G'_{best}$
16. **fin_si**
17. **para** $j = 1$ to TamPoblacion **hacer** /* para cada partícula */
18. **para** $k = 1\ to\ n$ **hacer** /* para cada palabra en la oración */
19. **si** $(Activo[j] = true)$ **entonces** /*¿la palabra tiene más de una etiqueta? */
20. $aleatorio \leftarrow r$ /* número aleatorio entre 0 y 1*/
21. **si** $(aleatorio < W)$ **entonces**
22. $P_k \leftarrow LB_k + r \times (UB_k - LB_k)$ /*selección de una etiqueta al azar */
23. **fin_si**
24. **sino** $(aleatorio < W + C1)$ **entonces**
25. $P_k \leftarrow P_{best\,(k)}$ /* se selecciona la etiqueta del mejor histórico */
26. **fin_si**
27. **sino** $(aleatorio < W + C1 + C2)$ **entonces**
28. $P_k \leftarrow G_{best\,(k)}$ /* se selecciona la etiqueta del mejor Global */
29. **fin_si**
30. **sino**
31. $P_k \leftarrow P_k$ /* se mantiene la etiqueta */
32. **fin_si**
33. **fin_si**
34. **fin_para**
35. **fin_para**
36. **fin_para**
37. **retornar** G_{best}

Fuente: Propia

Consolidado el algoritmo adaptado **PSOTagger**, se procede a realizar las pruebas con el ajuste de parámetros en cada corpus, donde para los corpus IULA y Brown se realizaron pruebas preliminares con datasets pequeños, los mejores resultados para cada corpus fueron:

- **Corpus IULA:** se ejecutaron diez pruebas, donde, el mejor experimento fue el PSOTagger con los parámetros W=0.3, C1=0.15, C2=0.45 y P=0.1, con una precisión de 92.883.

- **Corpus Brown:** se ejecutaron las mismas diez pruebas, donde el mejor experimento fue el PSOTagger con los parámetros W=0.3, C1=0.15, C2=0.45 y P=0.1, con una precisión de 89.650.
- **Corpus Nasa Yuwe:** se ejecutaron las mismas cuatro pruebas, donde el mejor experimento fue el PSOTagger con los parámetros W=0.3, C1=0.2, C2=0.4 y P=0.1, con una precisión de 55,936

En la **Tabla 49**, se muestran los resultados de los experimentos realizados anteriormente, pero en este caso utilizando los datasets completos de cada corpus.

Corpus	Algoritmo	Número de oraciones	Precisión (%)	Desviación estándar
IULA	PSOTagger	42079	96,7738	5,7844
Brown	PSOTagger	52998	89,6933	7,0810
Nasa Yuwe	PSOTagger	175	60,0749	16.7585

Tabla 49. Resultados de mejores experimentos PSOTagger. Fuente: Propia

4.4 Adaptación del algoritmo Jaya

La adaptación del algoritmo Jaya al problema de etiquetado se realizó de la siguiente manera: 1) se implementó el algoritmo DJaya para problemas discretos, teniendo en cuenta la discretización de [83]; 2) la solución se representa como la solución mostrada en la **Figura 22.** A continuación, en el **Algoritmo 2**, se presenta el pseudocódigo de JayaTagger adaptado al problema de etiquetado.

Algoritmo 2. JayaTagger

```
1.  Definición de parámetros: TamPoblacion, MaxGeneraciones, P4
2.  Inicialización aleatoria de la Población
3.  Definición de la mejor solución en la población (X_B)
4.  Definición de la peor solución en la población (X_W)
5.  para i = 1 to MaxGeneraciones hacer
6.      para j = 1 to TamPoblacion hacer /* para cada partícula */
7.          X_new /* generar una nueva solución*/
8.          para h = 1 to n hacer /* para cada palabra en la oración */
9.              si (Activo[h] = true) entonces /* ¿la palabra actual tiene más de
                                                  una posible
                                                  etiqueta? */
10.                 aleatorio ← r(1, P4)
11.                 según (aleatorio)
12.                   caso 1: /*preserve la etiqueta de la solución*/
13.                     X_new(h) = P_j(h)
14.                   caso 2: /*Seleccione la etiqueta de la mejor solución*/
15.                     X_new(h) = X_B(h)
16.                   caso 3: /*Seleccione la etiqueta aleatoriamente*/
17.                     X_new(h) = LB_h + r × (UB_h − LB_h)
18.                   caso 4: /*Seleccione la etiqueta de la peor solución*/
19.                     X_new(h) = X_W(h)
20.             fin_si
21.         fin_para
22.         Evaluar la función fitness de la nueva solución (X_new) (Figura 2)
23.         si (fitness (X_new) > fitness (P_j))
24.             P_j ← X_new
25.         fin_si
26.     fin_para
27. fin_para
28. retornar X_B /*Retorna solución de la población*/
```

Fuente: Propia

Consolidado el algoritmo adaptado **JayaTagger**, se procede a realizar las pruebas con el ajuste de parámetros en cada corpus, donde para los corpus IULA y Brown se realizaron pruebas preliminares con datasets pequeños, los mejores resultados para cada corpus fueron:

- **Corpus IULA:** se ejecutaron cuatro pruebas, el mejor experimento fue el JayaTagger con P4 igual a 4, con una precisión de 92.965.
- **Corpus Brown:** se ejecutaron las mismas cuatro pruebas, donde el mejor experimento fue el JayaTagger con P4 igual a 3, con una precisión de 92.916
- **Corpus Nasa Yuwe:** se ejecutaron las mismas cuatro pruebas, donde el mejor experimento fue el JayaTagger con P4 igual a 57.084

En la **Tabla 50**, se muestran los resultados de los experimentos realizados anteriormente, pero en este caso utilizando los datasets completos de cada corpus.

Tabla 50. Resultados de mejores experimentos JayaTagger. Fuente: Propia

Corpus	Algoritmo	Número de oraciones	Precisión (%)	Desviación estándar

IULA	JayaTagger	42079	96,8592	5,5963
Brown	JayaTagger	52998	89,6933	7,0810
Nasa Yuwe	JayaTagger	175	60,0749	16.7585

5. Algoritmo memético propuesto

En el artículo [90] se presentó una mejora del algoritmo GBHS, que maneja diferentes contextos de las palabras, lo que permitió obtener un mejor desempeño por el algoritmo GBHS2Pred2 (Contexto: dos palabras predecesoras y la palabra a etiquetar), incluso al trigrama presentado en [25] para el caso del castellano**.**

5.1 Adaptación del algoritmo Hill Climbing

La adaptación del algoritmo HC al problema de etiquetado, se realizó de la siguiente de la siguiente manera: 1) se utilizó una única solución donde se guarda las etiquetas de la oración a etiquetar; 2) la solución se representa como la solución mostrada en la **Figura 22;** 3) se utilizó una estrategia que selecciona entre dos métodos la palabra a etiquetar, el primer método, selecciona la palabra con la peor etiqueta en la solución y el segundo método, selecciona una palabra al azar sin importar la condición, evitando así, quedar atrapado en óptimos locales. en el **Algoritmo 3** se presenta el algoritmo HC adaptado al problema de etiquetado.

Algoritmo 3. HCTagger

1. Definición del parámetro $Prob$(Probabilidad para seleccionar la estrategia de selección de la palabra a etiquetar)
2. Definición del parámetro N (Número máximo de evaluaciones de la función objetivo)
3. Inicialización aleatoria de la $Solucion$
4. ListaPosibilidadesEtiquetas /* Esta lista contiene todas las posibilidades de las palabras a etiquetar*/
5. **para** $i = 1$ to N **hacer**
6. $aleatorio \leftarrow r(0,1)$ /* Se genera un numero aleatorio entre 0 y 1*/
7. $X_{new} \leftarrow$ Copiar$(Solucion)$
8. **si** $(aleatorio \leq Prob)$ **entonces** /*Si el número aleatorio es menor que $Prob$ definido en la configuración del experimento*/
9. $pos \leftarrow$ r(etiquetas) /* Se selecciona la posición de la palabra al azar */
10. **sino**
11. $pos \leftarrow$ indice de la palabra con la probabilidad mas baja /* Se selecciona la posición de la palabra basado en la probabilidad más baja
12. **fin_si**
13. $j \leftarrow LB_{pos} + r \times (UB_{pos} - LB_{pos})$ /*Se selecciona un índice al azar de las posibilidades de la palabra selecciona (pos)
14. $X_{new(pos)} =$ ListaPosibilidadesEtiquetas[pos][j] /*Se asigna la etiqueta a la nueva solución */
15. **si** (fitness (X_{new}) > fitness $(Solucion)$) **entonces**
16. $Solucion \leftarrow X_{new}$ /* Si la nueva solución supera a la actual se reemplaza */
17. **sino**
18. ListaPosibilidadesEtiquetas(pos).eliminar(j) /*sino se elimina la posibilidad de la lista*/
19. **fin_si**
20. **fin_para**
21. **retornar** $Solucion$

Fuente: Propia

Consolidado el algoritmo adaptado **HCTagger**, se realizaron las pruebas y el ajuste de parámetros, en este caso solo se realizó para el corpus IULA, con el fin de consolidar el algoritmo, para luego, integrarlo al algoritmo GBHSTagger como optimizador local, y de ahí si realizar las respectivas pruebas para los otros corpus.

5.2 Adaptación del algoritmo Hill Climbing con Reinicios Aleatorios (RRHC)

La adaptación del algoritmo RRHC al problema de etiquetado se realizó de la siguiente manera: 1) la solución se representa como la solución mostrada en la **Figura 22;** 2) la solución se reinicia (n_restart), seleccionando otra palabra de todas las posibilidades, luego esa palabra se mejora estocásticamente; 3) Se implementó una memoria tabú que guarda las palabras que fueron seleccionadas en el reinicio de la solución, esto con el fin, de no ser seleccionadas en una nueva solución, a menos que anteriormente una palabra vecina haya cambiado su etiqueta. A

continuación, en el **Algoritmo 4**, se presenta el pseudocodigo de **RRHCTagger** adaptado al problema de etiquetado.

Consolidado el algoritmo adaptado **RRHCTagger**, se procedió a realizar las pruebas con el ajuste de parámetros en cada corpus, donde para los corpus IULA y Brown se realizaron pruebas preliminares con datasets pequeños, los mejores resultados para cada corpus fueron:

- **Corpus IULA:** se ejecutaron cuatro pruebas, el mejor experimento fue el RRHCTagger con n_restart igual a 6, con una precisión de 94.111.
- **Corpus Brown:** se ejecutaron las mismas cuatro pruebas, donde el mejor experimento fue el RRHCTagger con n_restart igual a 4, con una precisión de 91.450
- **Corpus Nasa Yuwe:** se ejecutaron las mismas cuatro pruebas, donde el mejor experimento fue el RRHCTagger con n_restart igual a 63.709.

Algoritmo 4. RRHCTagger

```
1.  Definición del parámetro S, iterations_HC, i_restart y n_restart
2.  i_restart = 0
3.  Prob    /* Lista de las probabilidades de una palabra de acuerdo a todas las opciones */
4.  Inicialización aleatoria de S
5.  ActiveIndex     /* Esta lista guarda las posiciones de las palabras que tienen más de
                       una etiqueta*/
6.  EstadoTrigama       /* Esta lista guarda la posición de las palabras que fueron
                           seleccionadas*/
7.  j = Aleatorio(ActiveIndex.lenght)       /* j va a ser una posición al azar de activeIndex*/
8.  Prob = posiblesProbabilidades(j)        /*obtiene las posibles probabilidades de j*/
9.  X_new ← Copiar(S)
10. para i = 1 to iterations_HC  hacer
11.         si (i_restart > n_restart )             /*Si i_restart supera a n_restart */
12.             EstadoTrigama.add(j)
13.             si (fitness(X_new ) > fitness(S))
14.                 S ← X_new
15.                 si (EstadoTrigrama.contains(j − 1)) /*Si el vecino de j se
                                                          encuentra*/
16.                     EstadoTrigrama.remove(j − 1)/*Se elimina de la
                                                      lista */
17.                 fin_si
18.                 si (EstadoTrigrama.contains(j + 1))     /* Si el vecino
                                                               sucesor se encuentra */
19.                     EstadoTrigrama.remove(j + 1)    /* Se elimina de la
                                                           lista */
20.                 fin_si
21.                 X_new ← Copiar(S)       /*Se toma una nueva copia para aplicar la
                                               búsqueda estocástica*/
22.                 pos ← Aleatorio(ActiveIndex.lenght)     /* se selecciona una
                                                               nueva posición de activeIndex
23.                 j = ActiveIndex(pos)    /*La nueva palabra no debe estar en la
                                               lista de EstadoTrigrama */
24.                 Prob = posiblesProbabilidades(j)        /*Se obtienen las
                                                              probabilidades de la nueva palabra*/
25.                 i_restart = 0
26.             fin_si
27.             Prob.ordenar()          /*Se ordena de forma ascendente, con la mejor
                                          probabilidad en la primera posición */
28.             X2_new ← Copiar(X_new )         /* Se crea una nueva copia para las
                                                   modificaciones que se van a realizar*/
29.             pos2 =Prob[0]
30.             X2_new (j) = pos2               /*Se cambia la etiqueta en la posición j*/
31.             si (fitness(X2_new ) > fitness(X_new ) >))
32.                 X_new ← X2_new
33.                 Prob.remove(Prob[0]) /* Se elimina esta etiqueta, para no ser
                                            utilizada de nuevo*/
34.             else
35.                 Prob.remove(Prob[0])
36.                 i_restart + +
37.             fin_si
38. fin_para
39. retornar S
```

Consolidado el algoritmo adaptado **RRHCTagger**, En la sección 7 se presenta los resultados de este algoritmo con los corpus completos IULA, Brown y Nasa Yuwe.

5.3 Nueva versión del memético

La nueva versión memética, está basada en el algoritmo metaheurístico GBHS como optimizador local y HC como optimizador local. La versión de HC es la que se mostró

en la sección 5.1. En el algoritmo 5 se presenta el algoritmo GBHS adaptado al problema de etiquetado, la armonía se presenta como se muestra en la **Figura 22**, el optimizador local HC se adaptó a GBHS.

Consolidado el algoritmo, el cual fue nombrado **GBHS4Tagger**, se procedió a realizar las pruebas con el ajuste de parámetros en cada corpus, para los corpus IULA y Brown se realizaron pruebas preliminares con datasets pequeños, los mejores resultados para cada corpus fueron:

- **Corpus IULA:** se ejecutaron tres pruebas, donde el mejor experimento fue el GBHS4Tagger con Prob igual a 0.7, con una precisión de 94.314.
- **Corpus Brown:** se ejecutaron las mismas tres pruebas, donde el mejor experimento fue el GBHS4Tagger con Prob igual a 0.7, con una precisión de 91.283
- **Corpus Nasa Yuwe:** se ejecutaron las mismas tres pruebas, donde el mejor experimento fue el GBHS4Tagger con Prob igual a 0.75, con una precisión de igual a 61.418.

Consolidado el algoritmo adaptado **GBHS4Tagger**, En la sección 7 se presenta los resultados de este algoritmo con los corpus completos IULA, Brown y Nasa Yuwe.

Algoritmo 5. GBHS adaptado al problema de etiquetado

1. Definir parámetros: HMS, NI, HCMR, PARMin, ParMax, ProbOpt, Alpha, MaxNeighbors, Prob
2. Inicialización aleatoria de HM o inicialización mejorada usando el parámetro Alpha
3. **para** $i = 1$ to NI **hacer**
4. $PAR \leftarrow PARMin + (PARMax - PARMin) \times \left({}^{i}/_{NI}\right)$ /* definición de PAR */
5. **para** $j = 1$ to n **hacer** /* para cada palabra en la oración */
6. **si** (Activo[j] == true) **entonces** /* ¿la palabra actual tiene más de una posible etiqueta? */
7. **si** $(U(0,1) \leq HMCR)$ **entonces** /* memory consideration */
8. $x_j^{'} \leftarrow x_j^{p}$, donde $p \sim U(1, \dots, HMS)$
9. **Si** $(U(0,1) \leq PAR$ **entonces**
10. $x_j^{'} \leftarrow x_k^{best}$, donde best es el índice de la mejor armonía en HM and $k \sim U(1, n)$
fin_si
11. **sino** /* selección aleatoria */
12. $x_j^{'} \leftarrow LB_j + r \times (UB_j - LB_j)$
13. **fin_si**
14. **sino**
15. $x_j^{'} \leftarrow$ UnicaEtiquetaParaLaPalabra(j)
16. **fin_si**
17. **fin_para**
18. **mientras** (visitado ($x_j^{'}$)) **hacer** /* Si la solución ha sido visitada antes */
19. **si** (Activo[j] == true) **entonces /*** mutar la nueva armonía ($x_j^{'}$)
20. Se selecciona aleatoriamente una diferente a la actual
21. **fin_mientras**
22. Evaluar la función fitness de la nueva armonía ($x_j^{'}$)
23. **si** (fitness ($x_j^{'}$) > fitness de la peor armonía en HS) **entonces**
24. HM[pos_peor] $\leftarrow x_j^{'}$ /* reemplazar la peor en la memoria armónica */
25. **fin_si**
26. **si** $(U(0,1) < ProbOpt)$ **entonces**
27. Aplicar Optimización Local a la mejor armonía en HM
28. **fin_si**
29. **fin_para**
30. **retornar** la mejor armonía en HM

6. Resultados de los experimentos realizados sobre los mejores algoritmos metaheurísticos

6.1 Experimentos realizados con el corpus IULA

Los experimentos para el corpus IULA se ejecutaron con los algoritmos metaheurísticos GBHS4Tagger, RRHCTagger y GBHS2Pred2, utilizando el data-set completo. Cada algoritmo con su mejor versión, resultado de los experimentos preliminares. En la **Tabla 51**, se presentan los resultados de los experimentos realizados, además se replicó el experimento del etiquetado GBHS Tagger 2, presentado en [25].

Tabla 51. Resultados de mejores experimentos IULA. Fuente: Propia

Algoritmo	Frases	Precisión	Desviación estándar	Precisión (%) palabras desconocidas	Desviación estándar
GBHS4Tagger	42079	97.5403	5.4795	94.1986	7.4530
RRHCTagger	42079	97.2678	5.5006	94.0865	7.4208
GBHS2Pred2	42079	97.6051	5.6494	94.4238	7.8071
GBHS Tagger2	42079	97,4306	5,7844	94.1928	7.8055

En la **Tabla 51**, se puede observar que el algoritmo GBHS2Pred2 superó a los demás algoritmos en valor de precisión. Además de los resultados presentados en la **Tabla 51**, se aplicaron dos tests estadísticos, que nos permitieron ver las diferencias entre los algoritmos. En las dos siguientes secciones se muestran los resultados de cada test.

1. Test de Friedman

En la **Tabla 52,** se muestran los puntajes obtenidos para cada algoritmo, según el test de Friedman NXN, el algoritmo de etiquetado GBHS2Pred2 se posiciona como el mejor para el corpus IULA. Este test tiene 3 grados de libertad (15) y un valor de p (p-value) de 0.0018, que lo hace estadísticamente significativo.

Tabla 52. Resultados Test de Friedman – Corpus IULA. **Fuente:** Propia de software KEEL [79]

Algoritmo	Ranking Friedman
GBHS2Pred2 0.5 0.5	1
GBHS4Tagger 0.5 0.5 0.7	2
GBHS Tagger2 0.5 0.5	3
RRHCTagger 0.5 6	4

2. Test de Wilconson

La aplicación del test mostro con un 90% de significancia que los resultados obtenidos con el algoritmo GBHS2Pred2 son mejores que los otros algoritmos presentados en la **Tabla 51**, adicionalmente se pudo observar que el algoritmo GBHS4Tagger supera a el algoritmo GBHS Tagger 2.

6.2 Experimentos realizados con el corpus Brown

Los experimentos para el corpus Brown se ejecutaron con los algoritmos metaheurísticos GBHS4Tagger, RRHCTagger, utilizando el data-set completo. Cada algoritmo con su mejor versión, resultado de los experimentos preliminares. En la

Tabla 53, se presentan los resultados de los experimentos realizados, además se replicó el experimento del etiquetado GBHS Tagger 2, presentado en [25].

Tabla 53. Resultados de mejores experimentos Brown. Fuente: Propia

Algoritmo	Frases	Precisión	Desviación estándar	Precisión(%) palabras desconocidas	Desviación estándar
GBHS4Tagger	**52998**	**94.8803**	**6.0289**	**92.4432**	**6.3016**
RRHCTagger	52998	94.5342	6.0106	92.1588	6.3228
GBHS Tagger2	52998	94.8236	6.2051	92.3196	6.3888

En la

Tabla 53, se puede observar que el algoritmo GBHS4Tagger superó a los demás algoritmos en valor de precisión. Además de los resultados presentados en la

Tabla 53, se aplicaron dos tests estadísticos, que nos permitieron ver las diferencias entre los algoritmos. En las dos siguientes secciones se muestran los resultados de cada test.

1. Test de Friedman

En la **Tabla 54,** se muestran los puntajes obtenidos para cada algoritmo, según el test de Friedman NXN, el algoritmo de etiquetado GBHS4Tagger se posiciona como el mejor para el corpus Brown. Este test tiene 2 grados de libertad (10) y un valor de p (p-value) de 0.0067, que lo hace estadísticamente significativo.

Tabla 54. Resultados Test de Friedman – Corpus Brown. Fuente: Propia de software KEEL [79]

Algoritmo	Ranking Friedman
GBHS4Tagger 0.5 0.5 0.7	1
GBHS Tagger 2 0.5 0.5	2
RRHC Tagger 0.5 6	3

3. Test de Wilconson

La aplicación del test mostro con un 90% de significancia que los resultados obtenidos con el algoritmo GBHS4Tagger son mejores que los otros algoritmos presentados en la **Tabla 54.**

6.3 Experimentos realizados con el corpus Nasa Yuwe

Los experimentos para el corpus Brown se ejecutaron con los algoritmos metaheurísticos GBHS4Tagger, RRHCTagger, utilizando el data-set completo. Cada algoritmo con su mejor versión, resultado de los experimentos preliminares. En la **Tabla 55**, se presentan los resultados de los experimentos realizados, además se replicó el experimento del etiquetado GBHS Tagger 2, presentado en [25].

Tabla 55. Resultados de mejores experimentos Brown. Fuente: Propia

Algoritmo	Frases	Precisión	Desviación estándar	Precisión (%) palabras desconocidas	Desviación estándar
GBHS4Tagger	175	61.4185	15.5199	62.4022	14.7173
RRHCTagger	**175**	**63.7096**	**16.2249**	**63.6688**	**15.5570**
GBHS Tagger2	175	60.0749	15.5199	61.2142	15.9936

En la **Tabla 55**, se puede observar que el algoritmo RRHCTagger superó a los demás algoritmos en valor de precisión. Además de los resultados presentados en la **Tabla 55**, se aplicaron dos tests estadísticos, que nos permitieron ver las diferencias entre los algoritmos. En las dos siguientes secciones se muestran los resultados de cada test.

1. Test de Friedman

En la **Tabla 56,** se muestran los puntajes obtenidos para cada algoritmo, según el test de Friedman NXN, el algoritmo de etiquetado RRHCTagger se posiciona como el mejor para el corpus Brown. Este test tiene 2 grados de libertad (24.4914) y un valor de p (p-value) de 4.805692676468354E-6, que lo hace estadísticamente significativo.

Tabla 56. Resultados Test de Friedman – Corpus Nasa Yuwe. Fuente: Propia de software KEEL [79]

Algoritmo	Ranking Friedman
RRHCTagger 0.5 4	1
GBHS4Tagger 2 0.5 0.5 0.7	2
GBHS Tagger2 0.5 0.5	3

2. **Test de Wilconson:** la aplicación del test mostro con un 90% de significancia que los resultados obtenidos con el algoritmo RRHCTagger son mejores que los otros algoritmos presentados en la **Tabla 56**, adicionalmente se pudo observar que el algoritmo GBHS4Tagger supera a el algoritmo GBHS Tagger.

7. Prototipo Software para etiquetado de los corpus

Como último resultado de este trabajo se desarrolló un servicio web y una aplicación web que permita realizar el etiquetado de partes de discurso con los algoritmos presentado en este trabajo, utilizando la metodología Scrum. Para el desarrollo de estos dos artefactos se estableció una pila del producto (Product Backlog), que desencadeno la definición de 2 Sprint, cada uno con sus respectivas historias de usuario y criterios de validación.

El sprint 1 se enfocó en el desarrollo del servicio web, para la cual se utilizó una arquitectura API REST [86]. La codificación se realizó utilizando el framework ASP.NET, se realizaron las respectivos request y la terminación del sprint. En la **Figura 12**, se muestra un request que ejecuta el etiquetado.

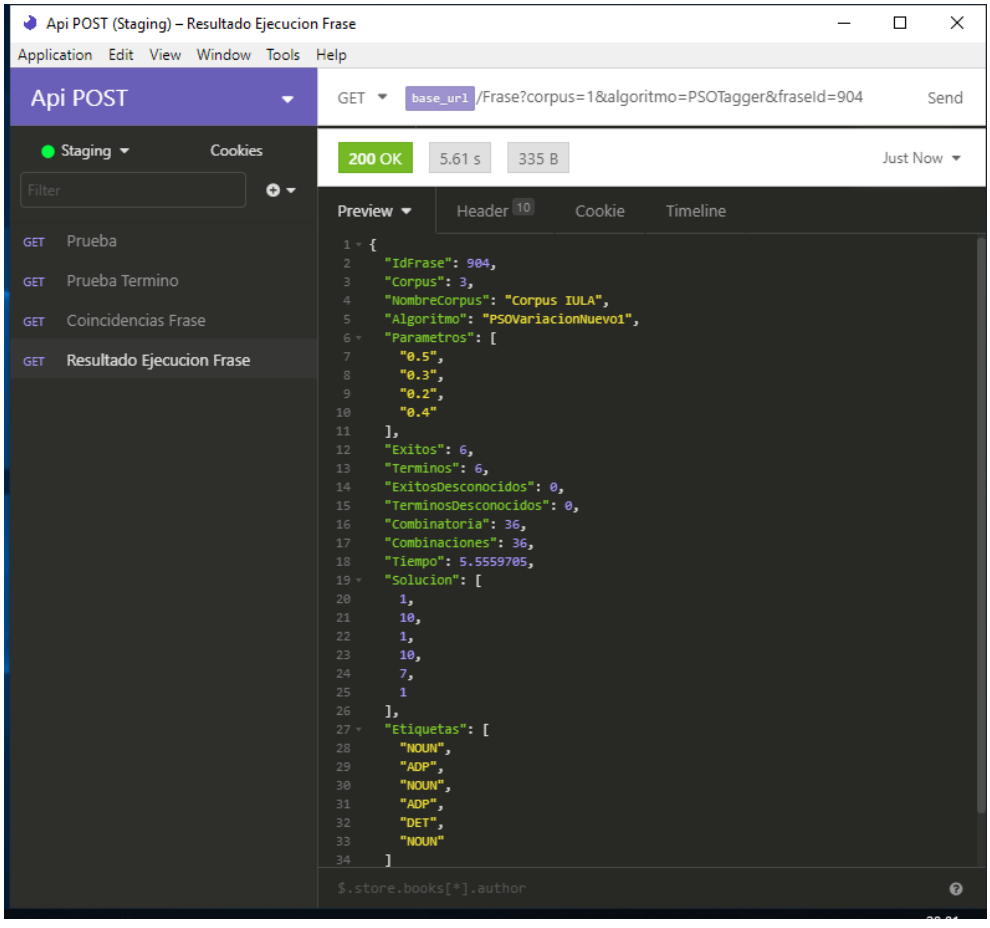


Figura 23. Request 4. Fuente: Propia

El sprint 2 se enfocó en el desarrollo de una aplicación web que consumiera el servicio web anteriormente desarrollado. La aplicación se desarrolló bajo el lenguaje Javascript

utilizando la librera React js [88]. Se realizaron las pruebas de integración con el API y se dio por terminado el sprint 2. En la **Figura 17**, se muestra la aplicación web, con el resultado de la ejecución del etiquetado.

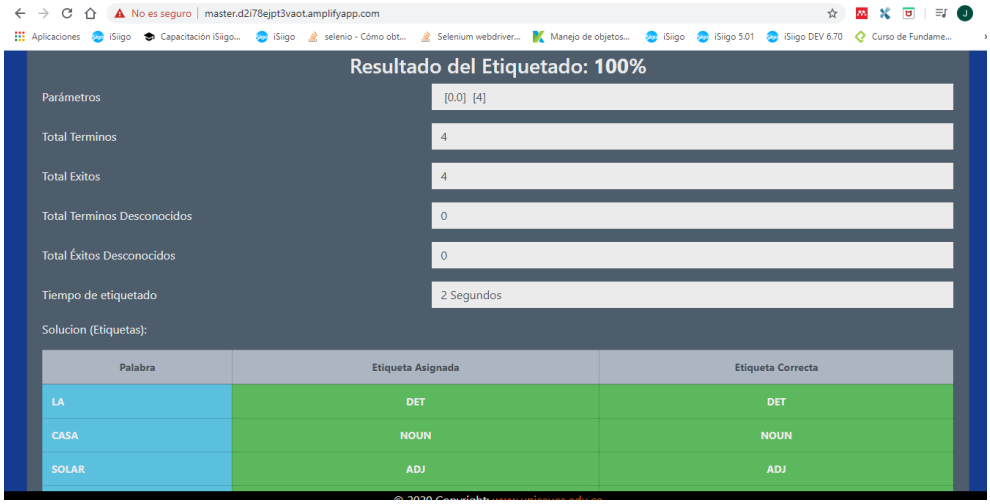


Figura 24. Resultado de etiquetado de un algoritmo en la app. Fuente: Propia.

Como última actividad se realizó el proceso de despliegue de los dos artefactos mencionados anteriormente. El servicio web se encuentra alojado en una máquina virtual, en la url **https://posttesis.southcentralus.cloudapp.azure.com/** y la aplicación web en un servicio app en la url **https://master.d2i78ejpt3vaot.amplifyapp.com/**. Para la ejecución de la aplicación web, se debe ir a la url del servicio web y permitir el contenido no seguro.

8. Conclusiones

El corpus IULA cuenta con un gran volumen de datos, lo cual permitió realizar validación cruzada de 5 folders favoreciendo la independencia entre los datos de entrenamiento y prueba, logrando que los algoritmos tengan una mayor capacidad de aprendizaje y se pueda enfrentar a situaciones en las cuales no tiene conocimiento.

este trabajo logró la adaptación de los algoritmos metaheurísticos Jaya [27], PSO [30] y GBHS [25] al problema de etiquetado, teniendo en cuenta las características propias de cada algoritmo y realizando el ajuste de parámetros requerido para cada algoritmo sobre cada corpus.

9. Referencias

[1] G. G. Chowdhury, “Natural language processing,” *Annu. Rev. Inf. Sci. Technol.*, vol. 37, no. 1, pp. 51–89, Jan. 2005, doi: 10.1002/aris.1440370103.

[2] A. Ekbal and S. Saha, “Simulated annealing based classifier ensemble

techniques: Application to part of speech tagging," *Inf. Fusion*, vol. 14, no. 3, pp. 288–300, 2013, doi: 10.1016/j.inffus.2012.06.002.

[3] K. Ravi and V. Ravi, "A survey on opinion mining and sentiment analysis: Tasks, approaches and applications," *Knowledge-Based Syst.*, vol. 89, no. C, pp. 14–46, 2015, doi: 10.1016/j.knosys.2015.06.015.

[4] M. Bordoloi and S. K. Biswas, "Graph-Based Sentiment Analysis Model for E-Commerce Websites' Data," *Cogn. Informatics Soft Comput. Proceeding CISC 2017*, pp. 453–462, 2019, doi: 10.1007/978-981-13-0617-4.

[5] J. Ma, H. Liu, D. Huang, and W. Sheng, "An English part-of-speech tagger for machine translation in business domain," *7th Int. Conf. Nat. Lang. Process. Knowl. Eng.*, pp. 183–189, 2011, doi: 10.1109/NLPKE.2011.6138191.

[6] S. Huet, G. Gravier, and P. Sébillot, "Morphosyntactic resources for automatic speech recognition," *6th Int. Conf. Lang. Resour. Eval. Lr. 2008*, pp. 692–698, 2008.

[7] R. Karimpour *et al.*, "Improving persian information retrieval systems using stemming and part of speech tagging," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5706, pp. 89–96, 2009, doi: 10.1007/978-3-642-04447-2_10.

[8] T. Güngör, *Handbook of Natural Language Processing (second edition)*. 2011.

[9] D. Jurafsky and J. H. Martin, "Speech and Language Processing," *Speech Lang. Process. An Introd. to Nat. Lang. Process. Comput. Linguist. Speech Recognit.*, vol. 21, pp. 151–176, 2009, doi: 10.1162/089120100750105975.

[10] E. Brill, "A simple rule-based part of speech tagger," *Proc. third Conf. Appl. Nat. Lang. Process.*, 1992, doi: 10.3115/974499.974526.

[11] D. Q. Nguyen, D. Q. Nguyen, D. D. Pham, and S. B. Pham, "A robust transformation-based learning approach using ripple down rules for part-of-speech tagging," *AI Commun.*, vol. 29, no. 3, pp. 409–422, 2016, doi: 10.3233/AIC-150698.

[12] L. Araujo, "Symbiosis of Evolutionary Techniques and Statistical Natural Language Processing," *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 14–27, 2004, doi: 10.1109/TEVC.2003.818189.

[13] A. P. Silva, A. Silva, and I. Rodrigues, *Part-of-speech tagging using evolutionary computation*, vol. 512.Spring. 2014.

[14] A. Paul, B. S. Purkayastha, and S. Sarkar, "Hidden Markov Model based Part of Speech Tagging for Nepali language," *undefined*, 2015.

[15] A. Ratnaparkhi, "A Maximum Entropy Model for Part-Of-Speech Tagging," *Proc. Conf. Empir. Methods Nat. Lang. Process. EMNLP-96*, pp. 133–142, 1996, doi: 10.1007/s00280-013-2178-x.

[16] T. Brants, "A Statistical Part-of-Speech Tagger," vol. 5, pp. 1–7, 2000.

[17] D. Surendran and G.-A. Levow, “Dialog Act Tagging with Support Vector Machines and Hidden Markov Models,” *Interspeech 2006 9th Int. Conf. Spok. Lang. Process.*, pp. 1950–1953, 2006.

[18] H. Schmid, “Part-of-speech tagging with Neural Networks,” *Eur. J. Cancer Prev.*, vol. 27, no. 4, pp. 296–302, 1994, doi: 10.1097/CEJ.0000000000000354.

[19] L. Araujo, G. Luque, and E. Alba, “Metaheuristics for Natural Language Tagging,” pp. 889–900, 2004, doi: 10.1007/978-3-540-24854-5_90.

[20] W. N. Francis and H. Kucera, “Brown Corpus Manual,” 1979. [Online]. Available: http://clu.uni.no/icame/manuals/BROWN/INDEX.HTM#bc8. [Accessed: 03-Dec-2018].

[21] A. P. Silva, A. Silva, and I. Rodrigues, “BioPOS : Biologically Inspired Algorithms for POS Tagging,” *Proc. First Int. Work. Optim. Tech. Hum. Lang. Technol.*, vol. 2, no. December, pp. 1–16, 2013.

[22] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, “Building a large annotated corpus of English: the penn treebank,” *Comput. Linguist.*, vol. 19, no. 2, pp. 313--330, 1993, doi: 10.1162/coli.2010.36.1.36100.

[23] “Mac-Morpho.” [Online]. Available: http://nilc.icmc.usp.br/macmorpho/. [Accessed: 05-Dec-2018].

[24] R. Forsati and M. Shamsfard, “Novel harmony search-based algorithms for part-of-speech tagging,” *Knowl. Inf. Syst.*, vol. 42, no. 3, pp. 709–736, 2014, doi: 10.1007/s10115-013-0719-6.

[25] L. M. Sierra Martínez, C. A. Cobos, and J. C. Corrales, “Memetic algorithm based on global-best harmony search and hill climbing for part of speech tagging,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10682 LNAI, pp. 198–211, 2017, doi: 10.1007/978-3-319-71928-3_20.

[26] L. M. Sierra Martínez, C. A. Cobos, C. J. Muñoz Corrales, T. Curieux Rojas, E. Herrera-viedma, and D. H. Peluffo-ordóñez, “Building a Nasa Yuwe Language Corpus and Tagging with a Metaheuristic Approach,” *19th Int. Conf. Intell. Text Process. Comput. Linguist.*, vol. 22, no. 3, pp. 881–894, 2018, doi: 10.13053/CyS-22-3-3018.

[27] P. Singh and H. Chaudhary, “A Modified Jaya Algorithm for Mixed-Variable Optimization Problems,” *J. Intell. Syst.*, vol. 0, no. 0, pp. 1–21, 2018, doi: 10.1515/jisys-2018-0273.

[28] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, “Metaheuristic research: a comprehensive survey,” *Artif. Intell. Rev.*, no. January, pp. 1–43, 2018, doi: 10.1007/s10462-017-9605-z.

[29] W. G. M. Wolpert, David H., “No Free Lunch Theorems for Optimization,” *Encycl. GIS*, vol. 1, no. 1, pp. 240–240, 2008, doi: 10.1109/4235.585893.

[30] F. Neri, E. Mininno, and G. Iacca, “Compact particle swarm optimization,” *Inf.*

Sci. (Ny)., vol. 239, pp. 96–121, 2013, doi: 10.1016/j.ins.2013.03.026.

[31] T. Rojas, “Desde arriba y por abajo construyendo el alfabeto nasa. La experiencia de la unificación del alfabeto de la lengua páez (nasa yuwe) en el Departamento del Cauca Colombia.,” *Construyendo El Alf.*, vol. 1, no. 1, p. 15, 2001.

[32] J. Brownlee, *Clever Algorithms*. 2011.

[33] S. Luke, *Essentials of Metaheuristics*. 2015.

[34] K. Schwaber and J. Sutherland, “La Guía de Scrum. La Guía Definitiva de Scrum: Las Reglas del Juego.,” p. 22, 2017.

[35] K. S. Pratt, “Design Patterns for Research Methods: Iterative Field Research,” *AAAI Spring Symp. Exp. Des. Real*, no. 1994, pp. 1–7, 2009.

[36] I. Project Management Institute, *Institute, A guide to the Project Management Body of Knowledge, 5 ed., Project Management Institute*. 2017.

[37] D. Cutting, J. Kupiec, J. Pedersen, and S. Penelope, “A Practical Part-of-Speech Tagger,” *Proc. Conf.*, pp. 133–140, 1992.

[38] R. Forsati, “Cooperation of Evolutionary and Statistical,” no. Aisp, pp. 550–555, 2012.

[39] I. Boussaïd, J. Lepagnot, and P. Siarry, “A survey on optimization metaheuristics,” *Inf. Sci. (Ny).*, vol. 237, pp. 82–117, Jul. 2013, doi: 10.1016/j.ins.2013.02.041.

[40] C. Cotta, “Una Visión General de los Algoritmos Meméticos.”

[41] S. Luke, *Essentials of Metaheuristics*. 2015.

[42] E. R. R. Kato, G. D. de A. Aranha, and R. H. Tsunaki, “A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing,” *Comput. Ind. Eng.*, vol. 125, pp. 178–189, Nov. 2018, doi: 10.1016/j.cie.2018.08.022.

[43] Y. Zhang, S. Wang, and G. Ji, “A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications,” *Mathematical Problems in Engineering*, vol. 2015. Hindawi Limited, 2015, doi: 10.1155/2015/931256.

[44] E. R. R. Kato, G. D. de A. Aranha, and R. H. Tsunaki, “A new approach to solve the flexible job shop problem based on a hybrid particle swarm optimization and Random-Restart Hill Climbing,” *Comput. Ind. Eng.*, vol. 125, pp. 178–189, Nov. 2018, doi: 10.1016/j.cie.2018.08.022.

[45] R. Venkata Rao, “Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems,” *Int. J. Ind. Eng. Comput.*, vol. 7, no. 1, pp. 19–34, Dec. 2016, doi: 10.5267/j.ijiec.2015.8.004.

[46] R. Kennedy, J., & Eberhart, “Particle swarm optimization,” *Proc. Proc. IEEE Int. Conf. neural networks*, vol. Vol. 4, pp. 1942–1948, 1995.

[47] M. Sevkli and A. R. Guner, “A Continuos Particle Swarm Optimization Algorithm for Uncapacited Facility Location Problem,” pp. 316–323, 2006.

[48] Q. Pan, M. F. Tasgetiren, and Y. Liang, “A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem,” vol. 35, pp. 2807–2839, 2008, doi: 10.1016/j.cor.2006.12.030.

[49] C. J. Liao, Chao-Tang Tseng, and P. Luarn, “A discrete version of particle swarm optimization for flowshop scheduling problems,” *Comput. Oper. Res.*, vol. 34, no. 10, pp. 3099–3111, Oct. 2007, doi: 10.1016/j.cor.2005.11.017.

[50] M. Clerc, “Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem,” Springer, Berlin, Heidelberg, 2004, pp. 219–239.

[51] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, “A novel particle swarm optimization algorithm with adaptive inertia weight,” *Appl. Soft Comput. J.*, vol. 11, no. 4, pp. 3658–3670, 2011, doi: 10.1016/j.asoc.2011.01.037.

[52] A. Abunaser, I. A. Doush, N. Mansour, and S. Alshattnawi, “Underwater image enhancement using particle swarm optimization,” *J. Intell. Syst.*, vol. 24, no. 1, pp. 99–115, Mar. 2015, doi: 10.1515/jisys-2014-0012.

[53] M. G. H. Omran and M. Mahdavi, “Global-best harmony search,” *Appl. Math. Comput.*, vol. 198, no. 2, pp. 643–656, 2008, doi: 10.1016/j.amc.2007.09.004.

[54] A. Alhasan and A. T. Al-taani, “POS Tagging for Arabic Text Using Bee Colony Algorithm,” *Procedia Comput. Sci.*, pp. 158–165, 2018, doi: 10.1016/j.procs.2018.10.471.

[55] M. El-Haj and R. Koulali, “KALIMAT a multipurpose Arabic Corpus,” *Second Work. Arab. Corpus Linguist.*, 2013.

[56] A. Al-Taani and S. A. Al-Rub, “A rule-based approach for tagging non-vocalized Arabic words,” *Int. Arab J. Inf. Technol.*, vol. 6, no. 3, pp. 320–328, 2009.

[57] S. Petrov, D. Das, and R. McDonald, “A Universal Part-of-Speech Tagset,” 2011, doi: 10.1038/hdy.2008.34.

[58] R. Forsati, M. Shamsfard, and P. Mojtahedpour, “An efficient meta heuristic algorithm for POS-tagging,” *Proc. - 5th Int. Multi-Conference Comput. Glob. Inf. Technol. ICCGI 2010*, pp. 93–98, 2010, doi: 10.1109/ICCGI.2010.42.

[59] E. Alba, G. Luque, and L. Araujo, “Natural language tagging with genetic algorithms,” *Inf. Process. Lett.*, vol. 100, no. 5, pp. 173–182, Dec. 2006, doi: 10.1016/J.IPL.2006.07.002.

[60] R. Forsati and M. Shamsfard, “Hybrid PoS-tagging: A cooperation of evolutionary and statistical approaches,” *Appl. Math. Model.*, vol. 38, no. 13, pp. 3193–3211, Jul. 2014, doi: 10.1016/J.APM.2013.11.047.

[61] A. P. Silva, A. Silva, and I. Rodrigues, “An Approach to the POS Tagging Problem Using Genetic Algorithms,” in *In Computation Intellingence*, 2012, pp. 3–17.

[62] A. P. Silva, A. Silva, and I. Rodrigues, “A New Approach to the POS Tagging Problem Using Evolutionary Computation,” *Proc. Recent Adv. Nat. Lang. Process.*, no. September, pp. 619–625, 2013.

[63] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, “A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA,” *IEEE Trans. Evol. Comput.*, vol. 12, no. 3, pp. 269–283, Jun. 2008, doi: 10.1109/TEVC.2007.900837.

[64] T. Curieux Rojas, “Esbozo gramatical de la lengua nasa (lengua páez),” *El Leng. en Colomb.*, vol. 1: Realida, no. Ed., Bogotá, Academia Colombiana de la Lengua e Instituto Caro y Cuervo, 2012.

[65] H. M. Alonso and D. Zeman, “Universal dependencies for the AnCora treebanks,” *Proces. Leng. Nat.*, vol. 57, pp. 91–98, 2016.

[66] Universal Dependencies, “Universal Dependencies.” [Online]. Available: http://universaldependencies.org/. [Accessed: 17-Dec-2018].

[67] J. Lavid, J. Arús, B. DeClerck, and V. Hoste, “Creation of a High-quality, Register-diversified Parallel (English-Spanish) Corpus for Linguistic and Computational Investigations,” *Procedia - Soc. Behav. Sci.*, vol. 198, pp. 249–256, Jul. 2015, doi: 10.1016/J.SBSPRO.2015.07.443.

[68] K. Toutanova, D. Klein, C. Manning, and Y. Singer, “Feautre-Rich Part-of-Speech Tagging with a Cyclic Dependency Network,” *Proc. 2003 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol.*, vol. 1, no. 4, pp. 173–180, 2003, doi: 10.1007/BF02379273.

[69] U. P. F. Institut Universitari de Lingüística Aplicada (IULA), “IULA Spanish LSP Treebank,” 2012.

[70] EAGLES, “ETIQUETAS EAGLES.” [Online]. Available: http://blade10.cs.upc.edu/freeling-old/doc/tagsets/tagset-es.html. [Accessed: 03-Dec-2018].

[71] I. Zeroual, A. Lakhouaja, and R. Belahbib, “Towards a standard Part of Speech tagset for the Arabic language,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 29, no. 2, pp. 171–178, 2017, doi: 10.1016/j.jksuci.2017.01.006.

[72] H. Schmid, “TreeTagger - a language independent part-of-speech tagger,” 1994. [Online]. Available: http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/treetagger.en.html. [Accessed: 12-Dec-2018].

[73] B. Kabashi and T. Proisl, “A Proposal for a Part-of-Speech Tagset for the Albanian Language,” *Proc. Tenth Int. Conf. Lang. Resour. Eval. (LREC 2016)*, pp. 4305–4310, 2016.

[74] M.-C. De Marneffe, B. MacCartney, and C. D. Manning, “Generating Typed Dependency Parses from Phrase Structure Parses,” *Proc. Lr.*, 2006, doi: 10.1145/1391729.1391731.

[75] D. Zeman, “Reusable Tagset Conversion Using Tagset Drivers,” *Proc. Lr.*, 2008.

[76] J. Lafferty, A. Mccallum, and F. Pereira, “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data Part of the Numerical Analysis and Scientific Computing Commons Recommended Citation "Conditional Random Fields: Probabilistic Models for Segmenting and Labelin,” *Proc. ICML*, vol. 2001, no. June, pp. 282–289, 2001.

[77] J. Tobar and M. Solano, “Dataset IULA,” 2020. .

[78] “slavpetrov/universal-pos-tags: Automatically exported from code.google.com/p/universal-pos-tags.” [Online]. Available: https://github.com/slavpetrov/universal-pos-tags. [Accessed: 27-Jun-2019].

[79] J. Alcalá-Fdez *et al.*, “KEEL: A software tool to assess evolutionary algorithms for data mining problems,” *Soft Comput.*, vol. 13, no. 3, pp. 307–318, 2009, doi: 10.1007/s00500-008-0323-y.

[80] S. Callander, “Searching and learning by trial and error,” *Am. Econ. Rev.*, vol. 101, no. 6, pp. 2277–2308, 2011, doi: 10.1257/aer.101.6.2277.

[81] A. K. Mishra and D. Shrivastava, “A discrete Jaya algorithm for permutation flow-shop scheduling problem,” vol. 11, pp. 1–14, 2020, doi: 10.5267/j.ijiec.2019.12.001.

[82] S. O. Degertekin, L. Lamberti, and I. B. Ugur, “Sizing, layout and topology design optimization of truss structures using the Jaya algorithm,” *Appl. Soft Comput. J.*, vol. 70, pp. 903–928, Sep. 2018, doi: 10.1016/j.asoc.2017.10.001.

[83] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan, “Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm,” *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1944–1955, 2019, doi: 10.1109/TCYB.2018.2817240.

[84] M. A. Al-Betar, “β-Hill climbing: an exploratory local search,” *Neural Comput. Appl.*, vol. 28, no. 1, pp. 153–168, Dec. 2017, doi: 10.1007/s00521-016-2328-2.

[85] C. Laguna, “Inferencia Paramétrica: Relación entre dos variables cualitativasy cuantitativas.,” *Inst. Aragon. ciencias la salud*, vol. 8, no. 2, p. 2,10, 2016.

[86] M. Masse, *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. 2011.

[87] “Insomnia REST Client.” [Online]. Available: https://insomnia.rest/. [Accessed: 31-Mar-2020].

[88] J. Walke, “React – Una biblioteca de JavaScript para construir interfaces de usuario.” [Online]. Available: https://es.reactjs.org/. [Accessed: 05-Apr-2020].

[89] L. M. Sierra, C. Cobos, and J. C. Corrales, “Continuous Optimization Based on a Hybridization of Differential Evolution with K-means,” *Bazzan A., Pichara K. Adv. Artif. Intell.*, vol. 8864, pp. 381–392, Nov. 2014, doi: 10.1007/978-3-319-12027-0_31.

[90] J. J. Tobar, M. A. Solano, L. M. Sierra, and C. A. Cobos, “Etiquetado de partes del discurso sobre un corpus en castellano basado en metaheurísticas,” *Rev. Ibérica Sist. e Tecnol. Inf.*, p. 15, 2020.

[91] D. Moussallem, M. Wauer, and A. N. Ngomo, “Web Semantics : Science , Services and Agents on the World Wide Web Machine Translation using Semantic Web Technologies : A Survey,” *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 51, pp. 1–19, 2018, doi: 10.1016/j.websem.2018.07.001.

[92] C. Garcia-Martinez, F. J. Rodriguez, and M. Lozano, “Ánalisis Empirico del No Free Lunch en Problemas Binarios Reales.” .