Algoritmo para Generación Automática de Resúmenes Extractivos Genéricos de un Documento basado en el Procedimiento de Búsqueda del Pescador

Andres Emiro Montilla Piamba

Trabajo de Grado para optar al título de Ingeniero de Sistemas

Director: Dr. (c) Martha Eliana Mendoza Becerra
Co-Director: Dr. Carlos Alberto Cobos

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información (GTI)
Línea Investigación: Gestión de la Información, Recuperación de la Información
Popayán
2016

Algoritmo para Generación Automática de Resúmenes Extractivos Genéricos de un Documento basado en el Procedimiento de Búsqueda del Pescador

Andres Emiro Montilla Piamba

Director: Dr. (c) Martha Eliana Mendoza Becerra
Co-Director: Dr. Carlos Alberto Cobos

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información (GTI)
Línea Investigación: Gestión de la Información, Recuperación de la Información
Popayán
2016

A nuestro Padre celestial, motor de nuestro existir, a nuestras familias, regalo maravilloso de Dios, y a todos aquellos que de una u otra forma hicieron suyos nuestros sueños.

Agradecimientos

Le agradezco a Dios por haberme acompañado y guiado a lo largo de mi carrera, por ser mi fortaleza en los momentos de debilidad y por brindarme una vida llena de aprendizajes, experiencias y sobre todo felicidad.

Le doy gracias a mis padres Nidia y Emiro por apoyarme en todo momento, por los valores que me han inculcado, y por haberme dado la oportunidad de tener una excelente educación en el transcurso de mi vida. Sobre todo por ser un excelente ejemplo de vida a seguir.

Martha Mendoza y Carlos Cobos, por darme la oportunidad de aprender y crecer en su grupo de trabajo, y a todos los profesores que me orientaron a lo largo de mi carrera para formarme como profesional.

CONTENIDO

Lista de tablas

Lista de figuras

Presentación

En la actualidad el gran volumen de información textual disponible en la web hace difícil que los usuarios puedan leer todos los documentos relevantes en una cierta temática y obtener las ideas más importantes contenidas en los mismos, debido al tiempo que esta tarea conlleva. En este contexto la generación automática de resúmenes extractivos es una herramienta importante que permite obtener de manera rápida y sencilla la información más relevante de cada documento.

En este documento, se propone un algoritmo que permite la generación automática de resúmenes de un documento basado en la metaheurística Procedimiento de Búsqueda del Pescador adaptando una memoria tabú Explícita (FSP-MT-SingleDocSum). A lo largo del documento se describen las bases teóricas y el proceso de desarrollo realizado.

En el capítulo 1 se presenta la descripción del problema que se aborda en este proyecto, la importancia del desarrollo del mismo, los objetivos trazados para plantear una alternativa de solución al problema planteado y los resultados obtenidos al finalizar el desarrollo de esta investigación.

El capítulo 2 presenta los conceptos más importantes del área de investigación de generación de resúmenes, los métodos del estado del arte en esta área, las medidas de calidad de resúmenes automáticos aceptadas por la comunidad científica. También, la representación de documentos en el modelo espacio vectorial y medidas asociadas para ponderación de términos, y la medida de similitud de cosenos. Además la descripción general de los algoritmos de Procedimiento de Búsqueda del Pescador (FSP), búsqueda local de Ascenso de la Colina, Temple Simulado y Búsqueda.

En el capítulo 3 se presenta el proceso llevado a cabo para el desarrollo del algoritmo propuesto FSP-MT-SingleDocSum, realizando la descripción de los cuatro ciclos realizados. En el primero contemplando la definición de la función objetivo y la adaptación del algoritmo FSP original al problema de generación automática de resúmenes, con su correspondiente configuración y afinamiento de parámetros. Los ciclos dos, tres y cuatro, enfocados en la adaptación de los algoritmos FSP con ascenso de colina, FSP con temple simulado y FSP con una memoria Tabú, respectivamente, junto con la configuración y afinamiento de parámetros de cada algoritmo.

En el capítulo 4 se presenta el algoritmo propuesto FSP-MT-SingleDocSum, realizando la descripción de los parámetros del algoritmo, la función objetivo, la representación de las soluciones y las modificaciones realizadas al algoritmo FSP original junto a la memoria tabú Explícita para su adaptación al problema de la generación automática de resúmenes de un solo documento.

En el capítulo 5 se presentan la arquitectura, diagrama de clases e interfaz del complemento en VS.NET para el programa ofimático Office Word, que permite generar resúmenes de documentos en idioma inglés.

En el capítulo 6 se describen las tareas de pre-procesamiento, los conjuntos de datos DUC2001 y DUC2002 y los resultados de la evaluación de las medidas ROUGE-1, ROUGE-2 y ROUGE-SU sobre esos conjuntos. Además se realiza una comparación utilizando los resultados obtenidos por los esquemas planteados en los cuatro ciclos de desarrollo del algoritmo propuesto FSP-MT-SingleDocSum, como resultados de publicaciones de algoritmos desarrollados por otros autores del estado del arte.

En el capítulo 7 se exponen las conclusiones de los procesos de desarrollo y evaluación del algoritmo propuesto FSP-MT-SingleDocSum. Asimismo, se incluyen posibles líneas de trabajo futuro que surgen de la presente investigación.

Al final se presentan las referencias bibliográficas utilizadas durante el desarrollo del proyecto.

Capítulo 1

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

En la actualidad el gran volumen de información textual disponible en la web hace difícil que los usuarios puedan leer todos los documentos relevantes en una cierta temática y obtener las ideas más importantes contenidas en los mismos, debido al tiempo que esta tarea conlleva. En este contexto la generación automática de resúmenes extractivos permite obtener un resumen con la información más relevante de cada documento. Algunas áreas de aplicación son: E-learning [1], para seleccionar la información más importante de un texto; Motores de búsqueda [2] para obtener un breve resumen del documento o página web, también para obtener un resumen basándose en información contextual del usuario [3]; E-mail, que resume las discusiones de correo electrónico o muestra un correo electrónico en dispositivos móviles con un tamaño reducido de pantalla [4].

El primer trabajo en esta área de investigación se realizó en la década de 1950 [5]. Desde entonces se han propuesto y evaluado una variedad de métodos que están clasificados [6-8] de acuerdo: al número de documentos, el resumen generado puede ser obtenido de un solo documento o de múltiples documentos; al propósito del resumen: indicativos que describen brevemente la idea principal del documento, informativos que buscan sustituir el documento brindando una versión abreviada del contenido y críticos que recogen el punto de vista del autor del resumen; a la audiencia a la que va dirigido: genéricos que dan igual importancia a los temas principales del documento, basado en consulta que tiene en cuenta la información más relevante según la consulta realizada por un usuario, enfocados en el usuario o en tópicos adaptados al usuario que son elaborados de acuerdo al perfil del lector; el lenguaje soportado: mono-lenguaje diseñados para un solo lenguaje y multi-lenguaje que dan soporte a múltiples lenguajes.

Además, teniendo en cuenta la forma cómo se obtiene el resumen [6, 7] estos pueden ser abstractivos o extractivos. En los abstractivos las oraciones del resumen no necesariamente están en el documento original y se enfocan en la coherencia del resumen, para lo cual usan herramientas de análisis lingüístico que requieren de memoria y capacidad de procesamiento debido a la complejidad de la tarea. De otra parte, los extractivos para formar el resumen seleccionan el conjunto de oraciones más relevantes del texto original, en general incluyendo las oraciones en el resumen en el mismo el orden en el que estaban en el documento original. En la actualidad las técnicas extractivas son más usadas debido a su simplicidad y tiempo de cómputo.

Estos sistemas extractivos han sido ampliamente estudiados y en la literatura se encuentran gran variedad de métodos para la generación automática de resúmenes para un documento, como: estadísticos [9, 10], basados en técnicas de reducción algebraica

[11-17], basados en técnicas de aprendizaje de máquina [18-26], basados en conectividad de textos [27-32], basados en grafos [33-38], basados en agrupamiento [39-41]. Otro grupo son los algoritmos basados en metaheurísticas como: algoritmos de búsqueda armónica [42], optimización de enjambres [43, 44], algoritmos genéticos [45-48], programación genética [49, 50], algoritmos meméticos [51], evolución diferencial [52, 53]. También técnicas híbridas que combinan dos o más métodos para la generación automática de resúmenes [54-58].

Los métodos basados en metaheurísticas abordan la generación de resúmenes como un problema de optimización global, buscando seleccionar el mejor conjunto de oraciones para conformar el resumen. Este enfoque ha obtenido buenos resultados mostrando que el uso de este tipo de algoritmos es un área prometedora de investigación [51].

De otro lado, una investigación reciente plantea una metaheurística, llamada Procedimiento de Búsqueda del Pescador (FSP, Fisherman Search Procedure) que explora nuevas soluciones usando una combinación equilibrada de: búsqueda global orientada a obtener diversidad en las soluciones candidatas iniciales y búsqueda local que explota con mayor profundidad en la vecindad de las soluciones candidatas [59], con el propósito de ofrecer una alternativa de solución a una gran variedad de problemas de optimización.

Considerando el potencial de los algoritmos basados en metaheurísticas, en este proyecto se plantea un algoritmo para la generación automática de resúmenes genéricos y extractivos de un solo documento, basado en el FSP y una memoria tabú explícita. Con este nuevo algoritmo se obtienen resúmenes de una mejor calidad (con respecto a medidas ROUGE, coincidencia de palabras entre el resumen generado automáticamente y el resumen generado por un humano) sobre los conjuntos de datos DUC2001 y DUC2002, comparado con métodos del estado del arte.

En este proyecto se planteó la siguiente pregunta de investigación: ¿Es posible obtener resúmenes extractivos genéricos de un solo documento con resultados similares o de mayor calidad a los establecidos por métodos metaheurísticos del estado del arte, usando un algoritmo de optimización discreta basado en el procedimiento de búsqueda del pescador y en un algoritmo de búsqueda local como Ascenso de colina, Enfriamiento Simulado o Búsqueda Tabú? Con los resultados obtenidos en este proyecto se logró dar una respuesta afirmativa a esta pregunta.

1.2 APORTES

Con el algoritmo propuesto se aporta nuevo conocimiento en el área de generación automática de resúmenes de un solo documento, específicamente en metaheurísticas con el procedimiento de búsqueda del pescador y una memoria tabú explícita en la generación de los puntos de la red (vecindario). Este conocimiento es de gran importancia para la comunidad de Sistemas Inteligentes y de Recuperación de la información.

Con el desarrollo de este proyecto se contribuye a la línea de investigación de Gestión de la Información y Sistemas Inteligentes del Grupo de I+D en Tecnologías de la Información,

específicamente en el desarrollo de una nueva solución informática basada en metaheurísticas que permita generar resúmenes automáticos de un solo documento con mayor calidad que otros métodos del estado del arte.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Proponer un algoritmo para la generación automática de resúmenes extractivos genéricos de un documento basado en el Procedimiento de Búsqueda del Pescador y un algoritmo de búsqueda local como Ascenso de la Colina, Temple Simulado o Búsqueda Tabú.

1.3.2 OBJETIVOS ESPECÍFICOS

- Modelar un algoritmo para la generación automática de resúmenes extractivos genéricos de un documento basado en el procedimiento de búsqueda del pescador y un algoritmo de búsqueda local como Ascenso de la Colina, Temple Simulado o Búsqueda Tabú, usando del patrón de investigación iterativa propuesto por Pratt en 2009 [60].

- Evaluar la calidad de los resúmenes generados con el algoritmo propuesto, utilizando documentos de noticias de la Conferencia de Entendimiento del Documento y medidas ROUGE[1], comparando los resultados obtenidos con los reportados por otros algoritmos del estado del arte.

- Desarrollar un complemento de Word que permita realizar la generación automática de resúmenes de documentos en inglés usando el algoritmo propuesto.

1.4 RESULTADOS OBTENIDOS

- **Monografía del trabajo de grado**, en la que se presentan los conceptos teóricos necesarios en el desarrollo del proyecto, el proceso de diseño y afinamiento de la función objetivo y el algoritmo propuesto FSP-MT-SingleDocSum, la evaluación de la calidad de los resúmenes generados y la comparación con otros algoritmos del estado del arte. Además se presenta la arquitectura, diagrama de clases e interfaz del complemento VS.NET para Word. Por último las conclusiones y el trabajo futuro que el GTI desarrollará con base en esta investigación.

- **Código del algoritmo propuesto**, junto con la especificación de su lógica y componentes.

- **Complemento de Word**, que permite añadir al programa Office Word la función de generar resúmenes automáticos del texto de documentos en idioma inglés usando el

[1] Las medidas ROUGE serán seleccionadas en el desarrollo del proyecto.

algoritmo propuesto FSP-MT-SingleDocSum. Además proporciona opciones para configurar los parámetros del FSP-MT-SingleDocSum, la función objetivo, los algoritmos de optimización local y otros específicos del problema.

- **Artículo**, que presenta la descripción del algoritmo propuesto FSP-MT-SingleDocSum, mostrando la representación de las soluciones, la configuración de sus parámetros, la función objetivo, y las modificaciones realizadas al algoritmo FSP original junto a la memoria tabú Explícita para su adaptación al problema de la generación automática de resúmenes de un solo documento. También se presentan los resultados obtenidos al evaluar los resúmenes generados, que se comparan con los mostrados por otros algoritmos del estado del arte. El artículo se envía a la revista internacional “Expert Systems with Applications” para su evaluación.

Capítulo 2

2 CONTEXTO TEÓRICO Y ESTADO DEL ARTE

En este capítulo se presentan los conceptos más importantes del área de investigación de generación de resúmenes, se realiza una revisión de los trabajos más relevantes en esta área, las medidas de calidad de resúmenes automáticos aceptadas por la comunidad científica. También, la representación de documentos en el modelo espacio vectorial y medidas asociadas para ponderación de términos, y la medida de similitud de cosenos. Además la descripción general del algoritmo Procedimiento de Búsqueda del Pescador (FSP) y de los algoritmos búsqueda local como Ascenso de la Colina, Temple Simulado y Búsqueda Tabú.

2.1 GENERACIÓN AUTOMÁTICA DE RESÚMENES DE TEXTO

La generación automática de resúmenes de textos es una tarea del procesamiento de lenguaje natural, que tiene por objetivo identificar el contenido más significativo de un documento o múltiples documentos para ser extraído en un resumen de tamaño corto, que represente el contenido del documento conservando su información importante [8].

En la literatura existen diversas formas de clasificar los resúmenes [8]:

- Según el número de documentos: el resumen puede ser obtenido de un documento o múltiples documentos.
- Según su propósito:
    - Indicativo: describe brevemente la idea principal del documento.
    - Informativo: buscan sustituir el documento brindando una versión abreviada del contenido.
    - Crítico: recoge el punto de vista del autor del resumen.
- Según a la audiencia a la que va dirigido:
    - Genérico: da igual importancia a los temas principales del documento y están destinados a un amplio grupo de usuarios.
    - Basados en consultas: el resumen recoge la información más relevante según la consulta realizada por un usuario.
    - Enfocados en el usuario o en tópicos: adaptados al usuario, son elaborados de acuerdo al perfil del lector.

También se pueden clasificar de acuerdo a la estrategia utilizada para extraer el resumen:

- Según la profundidad de procesamiento:
    - Estrategias poco profundas o superficiales: solo se utilizan características superficiales como frecuencia de términos, posición de palabras u oraciones, palabras clave, entre otras.
    - Estrategias profundas: utilizan técnicas avanzadas de procesamiento de lenguaje para modelar las entidades que aparecen en el texto y sus relaciones.
- Según la forma como el resumen es extraído:

- Técnicas abstractivas: usan conocimientos lingüísticos para generar el resumen mediante el análisis de la gramática y la semántica.
- Técnicas extractivas: extraen las palabras, oraciones o párrafos más relevantes del documento origen para ser incluidas en el resumen.

2.1.1 Métodos de generación automática de resúmenes de un solo documento

Existen numerosas investigaciones que proponen métodos de generación automática de resúmenes de un solo documento, entre ellos están, los métodos estadísticos, basados en: aprendizaje de máquina, conectividad de textos, grafos, técnicas de reducción algebraica, técnicas de agrupamiento y en metaheurísticas. A continuación se presentan las investigaciones más representativas de estos métodos.

2.1.1.1 Estadísticos

Las primeras investigaciones en el área de la generación automática de resúmenes extractivos para valorar cada oración e identificar las palabras clave u oraciones más relevantes utilizaron características estadísticas como: número de palabras clave, posición en el documento, similitud con el título, frecuencia de palabras u oraciones [9]. En general estos métodos realizan una selección de las palabras clave u oraciones, calculan el puntaje de cada oración de acuerdo a las características estadísticas escogidas y las oraciones con los mayores puntajes se incluyen en el resumen. La efectividad de las características estadísticas depende del formato y el estilo de escritura del documento, sin embargo estos métodos siguen siendo usados debido a su facilidad de implementación y menor complejidad computacional.

En [10] aplican un sistema que en tres niveles selecciona las mejores oraciones que formaran el resumen, en cada nivel evalúa y filtra las oraciones de acuerdo al puntaje obtenido según la característica respectiva al nivel en este orden: frecuencia de termino y frecuencia inversa de la oración, presencia de nombres de entidades y nombres propios.

2.1.1.2 Reducción algebraica

El enfoque de reducción algebraica más utilizado dentro de la generación automática de resúmenes de texto es el basado en Análisis Semántico Latente (LSA, Latent Semantic Analysis), que extrae, representa y compara significados de palabras mediante el análisis algebraico-estadístico de un texto, cuya hipótesis básica es que el significado de una palabra está determinado por su aparición frecuente junto a otras palabras. LSA aplica el algoritmo de descomposición de valores singulares sobre la matriz original del texto, que permite capturar las interrelaciones entre los términos, por lo que los términos y oraciones son agrupados sobre una base 'semántica' en lugar de sólo palabras [11, 13-16]. A diferencia de LSA [12, 17], la factorización de matrices no negativas usa componentes no negativos que son más similares al proceso cognitivo humano, mostrando mejores resultados que LSA.

2.1.1.3 Aprendizaje de máquina

Es un enfoque supervisado que requiere realizar un entrenamiento previo con un conjunto de datos que permita obtener las probabilidades o pesos optimizados de las características utilizando diversos clasificadores: bayesiano [18, 21, 24], redes neuronales artificiales [22, 26], modelos ocultos de Markov [25, 61], campos aleatorios condicionales [23], árboles de decisión o lógica difusa [19, 20]. Cada una de las palabras clave u

oraciones extraídas del documento se clasifican de acuerdo a la probabilidad dada por la evaluación de las características, seleccionando aquellas con la probabilidad más alta, para ser incluidas en el resumen [8]. Esto métodos tienen como desventaja la necesidad de datos de entrenamiento que son difíciles de encontrar, además generan una dependencia con el lenguaje en que están escritos los documentos de entrenamiento.

2.1.1.4 Conectividad del texto

Buscan identificar las relaciones entre conceptos del documento, utilizando estrategias como las cadenas léxicas y las estructuras retóricas. En los métodos basados en cadenas léxicas la extracción de las palabras candidatas del documento se determina usando un tesauro como WordNet [29, 32], luego se revisa palabra por palabra observando si se pueden incluir en una cadena léxica existente o de lo contrario crear una nueva cadena léxica y por último se identifican las cadenas más fuertes y se extraen las oraciones más significativas para conformar el resumen. Los métodos basados en estructuras retóricas [27, 31] realizan la extracción de segmentos retóricos del documento original para formar una estructura de árbol, donde las unidades de texto constituyen los nodos (clasificados como núcleo o satélite, según su grado de relevancia para el discurso), luego se establece el puntaje de cada estructura retórica de acuerdo a las métricas que defina el algoritmo y se seleccionan las estructuras que obtengan los mejores puntajes para ser incluidas en el resumen. Aunque este enfoque mantiene el resumen en su contexto y con cohesión, su precisión disminuye a medida que se incrementa la cantidad de texto, para superar esta limitación en [28] se presenta un modelo híbrido que combina estructuras retóricas y el modelo de espacio vectorial. Estos métodos presentan desventajas como el uso de técnicas complejas y la dependencia del lenguaje para el procesamiento del texto [8].

2.1.1.5 Grafos

Las palabras clave u oraciones son representadas como nodos en un grafo no dirigido, cuando dos oraciones son similares son conectadas con un arco que tiene un peso asociado que indica la fortaleza de la conexión. Así se crea un grafo que representa las relaciones entre todas las oraciones del documento que permite ser iterado hasta que converja a un criterio, para luego ordenar y seleccionar las oraciones [35, 38]. Adicionalmente [37] combina el peso de cuatro tipo de arcos (similitud, similitud semántica, Resolución de la correferencia y Relaciones del Discurso). A diferencia de los estudios mencionados con anterioridad en [36], las secuencias frecuentes máximas (MFS, Maximal Frequent Secuences) son representadas como nodos. Las MFS son los n-gramas frecuentes que no pertenecen a ningún otro n-grama frecuente, que al ser encontradas y seleccionadas ofrecen la información más importante de un documento. En [33] se aborda de manera unificada la generación automática de resúmenes de un documento y múltiples documentos con un algoritmo basado en grafos, en el resumen se incluyen los nodos u oraciones que obtiene las mejores puntuaciones. Estos métodos no supervisado tiene la ventaja de ser independiente del lenguaje y de mejorar la cohesión en los resúmenes generados, de otro lado su mayor desventaja radica en el aumento de la complejidad computacional a medida que el número de nodos y arcos del grafo se incrementa.

2.1.1.6 Agrupamiento

Este enfoque busca formar grupos de oraciones similares (temáticas), las agrupaciones con muchas oraciones representan los temas más importantes del documento y las oraciones que representen más cada grupo se seleccionan para formar el resumen, evitando redundancia en el mismo. En [39] utilizan el algoritmo de agrupamiento maximizando la expectativa (EM, Expectation Maximization), para descubrir los grupos de oraciones con significado similar, conformando el resumen con la oración más representativa de cada grupo. En este caso, EM representa cada oración del documento en el modelo de espacio vectorial, como un vector de características que corresponde a los diferentes términos en el documento (bolsa de palabras, n-gramas o de MFS).

2.1.1.7 Metaheurísticas

Los métodos basados en metaheurísticas han sido explorados aplicando distintas técnicas que han mostrado buenos resultados y se han utilizado de dos formas: (i) para calcular los pesos de las características presentes en una ecuación que mide la relevancia de cada oración respecto al documento original, en este caso [43, 46, 49, 50], las soluciones candidatas representan la combinación de pesos de las características, el objetivo es encontrar una combinación adecuada que permita evaluar la relevancia de cada oración en el documento y así incluir en el resumen aquellas con la mejor puntuación. (ii) para generar automáticamente el resumen [42, 44, 45, 47, 48, 51, 52, 62], abordando el problema de generación de resúmenes como un problema de optimización. A continuación se describen varias investigaciones que utilizan esta segunda forma.

Un algoritmo memético que combina un algoritmo genético y la búsqueda local guiada, fue presentado en [51], representando en cada agente el conjunto de oraciones que forman un resumen candidato, maximizando una función objetivo que evalúa cinco características para cada solución, al final el algoritmo obtiene la solución más óptima que contiene el conjunto de oraciones que conformaran el resumen.

Un algoritmo de búsqueda armónica binaria [42] define una función objetivo que evalúa las oraciones basándose en aspectos como: el grado de relación entre las oraciones consecutivas (legibilidad), la similitud entre las oraciones del resumen (cohesión) y la similitud de las oraciones con el título del documento (relación al tópico). Estos factores son ponderados para ajustar la función objetivo de acuerdo a la necesidad y las oraciones con mejor evaluación de aptitud conforman el resumen.

Un algoritmo de evolución diferencial [52] define una función objetivo para optimizar el proceso de agrupamiento de oraciones, teniendo en cuenta dos criterios a optimizar: la disimilitud entre las oraciones asignadas a cada grupo (inter-grupo) y la similitud entre las oraciones del mismo grupo (intra-grupo). Al final se obtienen los grupos de oraciones con el mayor valor de la función de aptitud y las mejores oraciones de cada grupo son escogidas para conformar el resumen.

Las metaheurísticas también han sido utilizadas en combinación con otras técnicas para la generación automática de resúmenes, buscando mejorar la calidad de las soluciones encontradas aprovechando las ventajas de cada técnica en las tareas utilizadas para generar el resumen. En [56] proponen usar: Autómatas Celulares de Aprendizaje (CLA , Cellular Learning Automata) para calcular la similitud de las oraciones, Optimización por Enjambre de Partículas (PSO, Particle Swarm Optimization) para asignar los pesos

apropiados a las características de acuerdo a su importancia y lógica difusa para puntuar las oraciones. Este estudio propone dos métodos, el primero combina CLA y PSO y el segundo CLA, PSO y Lógica Difusa; mostrando mejores resultados con el segundo método.

También Modelo de Optimización Evolutivo Difuso (FEOM, Fuzzy Evolutive Optimization Model) [54] es un método que combina los algoritmos los genéticos y la lógica difusa. Los primeros generan una población aleatoria como el grupo inicial de soluciones para el agrupamiento de oraciones y la lógica difusa para realizar de forma adaptativa las estrategias de evolución (selección, cruce y mutación) que prevengan una convergencia prematura a un óptimo local. Finalmente se seleccionan las oraciones más importantes de cada grupo para obtener el resumen.

Otro método hibrido [55] utiliza PSO para optimizar desde un conjunto de datos de entrenamiento los pesos de las características que definen la forma de evaluar a cada una de las oraciones del documento, combinado con un método basado en diversidad para filtrar las oraciones similares y seleccionar las más diversas y lógica difusa para mejorar la precisión. El método se presenta en dos formas: en la primera la diversidad domina el modelo y en la segunda la diversidad no se impone en el modelo; ésta última forma obtiene los mejores resultados.

2.1.2 Métodos de evaluación de la calidad de los resúmenes

La evaluación es un aspecto importante en el desarrollo de cualquier sistema o método, en el caso de los sistemas de generación automática de resúmenes, es una tarea compleja debido a las irregularidades de los lenguajes (humanos) y a que la concepción de lo que es un buen resumen varía entre las personas. En la actualidad no existe un esquema de evaluación definitivo, sin embargo, se han desarrollado diversas herramientas que permiten su automatización haciendo uso de las medidas estándar de recuperación de información como la precisión, recuerdo y medida F [63].

Los métodos para evaluar los sistemas de generación automática de resúmenes de texto, se dividen en dos grandes categorías intrínsecas y extrínsecas [64].

2.1.2.1 Evaluación intrínseca

La mayoría de los esquemas de evaluación de resúmenes son intrínsecos, este tipo de evaluación mide el sistema sin tener en cuenta su audiencia objetivo, y con frecuencia se lleva a cabo mediante la comparación con un conjunto de resúmenes ideales[2], que pueden ser generados por evaluadores humanos o sistemas de referencia. Sin embargo, debido a la subjetividad que acompaña la creación de un resumen, por cada documento original suele utilizarse un puntaje promedio de varios resúmenes ideales generados por humanos [64].

La evaluación intrínseca se ha centrado principalmente en la coherencia y la capacidad informativa de los resúmenes, midiendo de ese modo solamente la calidad de salida [64].

[2] Este tipo de conjuntos suele conocerse como gold-standard corpus, y usualmente son vistos como modelos de excelencia que representan el límite más alto al que razonablemente se puede llegar por medios automáticos. Dentro de éste trabajo, este tipo de resúmenes serán mencionados como resúmenes ideales, resúmenes modelo o resúmenes de referencia

De esta forma, surgen algunas medidas, utilizadas comúnmente en este tipo de evaluación, como la *precisión* y el *recuerdo* [65]. El recuerdo es la razón del número de oraciones comunes entre el resumen generado y el de referencia, sobre el número total de las oraciones del resumen generado. Análogamente, la precisión se define como el número de oraciones en el resumen generado que están presentes en el resumen de referencia. Precisión y la recuperación son medidas estándar para la recuperación de información y, a menudo se combinan en la denominada medida F.

En la actualidad el paquete de medidas de ROUGE [66] se ha utilizado como una forma automatizada de evaluación de resúmenes, que se basa en la cantidad de unidades comunes entre un resumen generado y un resumen ideal.

2.1.2.2 Evaluación extrínseca

Este tipo de evaluación se orientó hacia el usuario final. Por lo tanto, mide la eficacia y aceptabilidad de los resúmenes generados en alguna tarea, por ejemplo, la evaluación de la pertinencia o la comprensión de la lectura. Además, si el resumen contiene algún tipo de instrucciones, es posible medir hasta qué punto es posible seguir las instrucciones y el resultado del mismo. Otras posibles tareas medibles son la recopilación de información en una gran colección de documentos, el esfuerzo y el tiempo necesario para enviar a editar el resumen generado por una máquina con un propósito específico, o el impacto del sistema generación de resúmenes en un sistema del que forma parte, por ejemplo relevancia retroalimentación (ampliación de consultas) en un motor de búsqueda o un sistema de pregunta-respuesta [64].

2.1.2.3 Evaluación ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) es un paquete de software que utiliza un conjunto de medidas para determinar automáticamente la calidad de un resumen comparándolo con otros resúmenes ideales creados por humanos. Las medidas cuentan el número de unidades superpuestas, tales como n-gramas de palabras, y pares de palabras, entre el resumen generado por computador y los resúmenes ideales [66]. Así pues, la evaluación se lleva a cabo a través del conteo de unidades coincidentes entre los resúmenes.

El paquete de evaluación de ROUGE incluye varias medidas como ROUGE-N, ROUGE-L, ROUGE-W, ROUGE-S y ROUGE-SU. Entre las más usadas están ROUGE-1, ROUGE-2 y ROUGE-SU4.

- **ROUGE-N**

ROUGE-N es una medida basada en el recuerdo de n-gramas entre un resumen generado y un conjunto de resúmenes de referencia. En la Ecuación (2.1) se presenta el cálculo de esta medida.

$$\text{ROUGE} - \text{N} = \frac{\sum_{S \in \{ResúmenesDeReferencia\}} \sum_{grama_n \in S} Conteo_{Coincidencia}(grama_n)}{\sum_{S \in \{ResúmenesDeReferencia\}} \sum_{grama_n \in S} Conteo(grama_n)} \quad (2.1)$$

Donde n representa la longitud del n-grama ($grama_n$) y $Conteo_{Coincidencia}(grama_n)$ es el máximo número de n-gramas coincidentes entre un resumen candidato y un conjunto de resúmenes de referencia. El denominador de esta fórmula corresponde a la suma de la

cantidad de n-gramas en el resumen de referencia, de ahí que su valor crecerá conforme al número de resúmenes ideales. De esta manera, un resumen generado que comparta palabras con más de un resumen de referencia obtendrá un mejor valor para la medida ROUGE-N.

- **ROUGE-S**

ROUGE-S mide la superposición de saltos de bigramas (bigramas-skip) entre un resumen candidato y un conjunto de resúmenes de referencia. Un bigrama-skip se refiere a un par de palabras, en el orden en que están en la oración, permitiendo saltos arbitrariamente. Dadas una oración de referencia X, de longitud m, y una oración candidata Y, de longitud n, el cálculo de la medida-F basada en bigramas-skip corresponde al cálculo de ROUGE-S como se aprecia en las Ecuaciones (2.2),(2.3) y (2.4).

$$R_{skip2} = \frac{SKIP2(X,Y)}{C(m,2)} \qquad (2.2)$$

$$P_{skip2} = \frac{SKIP2(X,Y)}{C(n,2)} \qquad (2.3)$$

$$F_{skip2} = \frac{(1+\beta^2)R_{skip2}P_{skip2}}{R_{skip2}+\beta^2 P_{skip2}} \qquad (2.4)$$

Donde $SKIP2(X,Y)$ es la cantidad de bigramas-skip que coinciden entre X e Y, β se encarga de controlar la importancia relativa de P_{skip2} y R_{skip2}, y C es la función de combinación que calcula la cantidad de bigramas-skip presentes en una oración[3]. Considerando el siguiente ejemplo:

S_1: police killed the gunman
S_2: the gunman police killed

Se infiere que cada oración tiene 6 bigramas-skip[4]. Para S_1 los bigramas-skip corresponden a {"police killed", "police the", "police gunman", "killed the", "killed gunman", "the gunman"}. S_2 tiene dos bigramas-skip que coninciden con S_1 y son {"police killed", "the gunman"}. De esta forma, P_{skip2} y R_{skip2} entre S_1 y S_2 son igual a 0.3333, así ROUGE-S se calcula como 0.3333.

- **ROUGE-SU**

Un problema que presenta ROUGE-S es que no da ningún valor a una oración candidata si ésta no tiene ningún par de palabras coincidentes con otro par en las oraciones de referencia. ROUGE-SU evita este problema tomando como punto de partida ROUGE-S, pero incluyendo el manejo de unigramas como conteo de unidades. De esta manera, ROUGE-SU adiciona un marcador al inicio de las oraciones candidata y de referencia.

[3] La fórmula general para calcular las combinaciones que se pueden obtener con n elementos, tomados de r en r, es C(n,r) = (n!/r!*(n-r)!)
[4] C(4,2) = (4!/2!*2!) = 6

2.1.2.4 Colección de documentos de evaluación

La Conferencia de Entendimiento del Documento (DUC, Document Understanding Conference), define resúmenes "ideales" creados por humanos expertos sobre un conjunto de documentos originales, con el objetivo de ofrecer a la comunidad académica conjuntos de datos que permitan evaluar y comparar los resultados obtenidos por sistemas de generación automática de resúmenes. DUC tiene una gran aceptación por parte de la comunidad académico-científica, ya que su conjunto de documentos es frecuentemente utilizado como conjunto de referencia por diversos estudios [8, 51, 53].

2.2 REPRESENTACIÓN DE LOS DOCUMENTOS

2.2.1 Modelo de espacio vectorial

La representación de un conjunto de textos[5] como vectores se conoce como modelo de espacio vectorial y es usado en los sistemas de recuperación de información a inicios de los 70 [67], para llevar a cabo diversas tareas como: puntuación en una consulta, clasificación de documentos u oraciones, agrupamiento de documentos u oraciones, etc. Este modelo ha sido ampliamente utilizado en la generación automática de resúmenes, representando al documento por medio de un vector de pesos de términos. Un término, puede ser una palabra o una oración. Así mismo, cualquier texto puede representarse por medio de un vector dentro de ese espacio (Ver Figura 2.1). De esta forma, si un término pertenece a un texto, obtiene un valor de acuerdo a su importancia dentro del texto según la técnica de ponderación de términos utilizada [68].

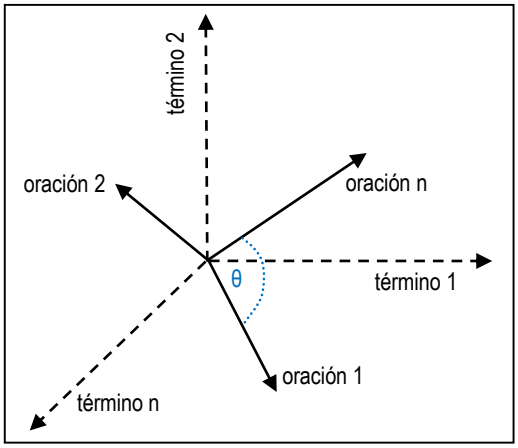

Figura 2.1. Representación Modelo Espacio Vectorial

Las coordenadas determinadas por los términos del texto lo sitúan dentro del espacio vectorial. Según este modelo, dado un texto T_i de n términos, su representación vectorial correspondería a $T_i = (w_{i1}, w_{i2}, w_{i3}, \dots, w_{in})$, donde w_{ij} es el peso, ponderación o importancia del término j en el texto i.

[5]Dependiendo de la aplicación, un texto puede ser una oración, un conjunto de oraciones o un documento.

Esta representación da como resultado la matriz de texto por términos (ver Figura 2.2), donde las filas representan los textos de la colección y las columnas representan cada uno de los términos existentes en cualquiera de los textos (un peso puede ser cero).

$$Matriz_{texto,\ termino} = \begin{Bmatrix} w_{1,1} & \ldots, & w_{1,n} \\ \ldots, & \ldots, & \ldots \\ w_{m,1} & \ldots, & w_{m,n} \end{Bmatrix}$$

Figura 2.2 Matriz de Textos por Términos

En esta matriz, se observan los vectores de pesos como proyecciones en un espacio multidimensional, donde la dimensión está dada por el número de textos m y el número de términos n. De este modo se puede medir la similitud léxica entre textos, mediante el cálculo de alguna medida de similitud. Por ejemplo, si queremos medir la similitud entre dos textos, los podemos representar con los siguientes vectores:

$$\overrightarrow{texto_i} = (w_{i,1},\ w_{i,2}, \ldots, w_{i,n})$$
$$\overrightarrow{texto_j} = (w_{j,1},\ w_{j,2}, \ldots, w_{j,n})$$

Figura 2.3 Representación de textos como vectores

En esta notación $\overrightarrow{Texto_i}$ y $\overrightarrow{Texto_j}$ son los vectores que representan los dos textos, respectivamente. De esta manera se puede calcular la similitud por medio de la medida Similitud Coseno (ver sección 2.2.3). Esta técnica es usada en la mayoría de estudios del estado del arte [42, 55, 69, 70].

2.2.2 Técnicas de ponderación de términos

En esta sección, se presentan algunas de las técnicas más utilizadas en la generación automática de resúmenes para la ponderación de términos.

2.2.2.1 Booleana

Este esquema binario concede la misma relevancia a cada término que aparece en el documento. Puede ser usado cuando no se considera importante el número de apariciones del término. El peso $w_{ij} \in \{0,1\}$ indica la ausencia o presencia del término j dentro del texto i. Se define como muestra la Ecuación (2.5) [71].

$$w_{ij} = \begin{cases} 1, & si\ el\ término\ j\ está\ en\ el\ texto\ i \\ 0, & en\ caso\ contrario \end{cases} \quad (2.5)$$

2.2.2.2 Frecuencia del término

Este esquema cuenta cuántas veces se utiliza el término en un documento. Si la frecuencia del término (TF, Term Frequent) en el documento es mayor, aumenta la probabilidad de que sea relevante para el documento. Como se observa en la Ecuación

(2.6), en este esquema la importancia de un término radica en la cantidad de veces que aparezca en el texto.

$$w_{ij} = f_{ij} \quad (2.6)$$

Donde f_{ij} es la frecuencia del término j en el texto i.

La desventaja de la aplicación de este método reside en que existen términos muy comunes que pueden aparecer en cualquier parte del texto sin, que por ello, contenga información relevante para caracterizarlo o diferenciarlo. Este tipo de términos tendrían una alta importancia aun cuando sean mucho menos representativos que otros. Así mismo, los términos pertenecientes a textos con mayor longitud tendrían mayor frecuencia que aquellos presentes en textos más cortos [72]. De esta forma, el cálculo de frecuencia más comúnmente utilizado, es el que se observa en la Ecuación (2.7).

$$w_{ij} = \frac{f_{ij}}{MáxFreq_i} \quad (2.7)$$

Donde $MáxFreq_i$ indica la cantidad de ocurrencias del término más frecuente dentro del texto i.

2.2.2.3 Frecuencia inversa de un término

Es un mecanismo para atenuar el efecto de los términos demasiado frecuentes en el texto, la idea es reducir el peso TF de un término con un factor que crece con su frecuencia de aparición. La frecuencia inversa de documento (IDF, Inverse document frequency), hace referencia a la frecuencia de un término dentro de una colección de textos [63]. La Ecuación (2.8) expresa esta definición.

$$w_{ij} = \log \frac{N}{n_j} \quad (2.8)$$

Donde N es la cantidad de textos de la colección y n_j es la cantidad de textos donde aparece el término j.

2.2.2.4 Frecuencia relativa de un término

Este esquema combina la definición de los métodos descritos en las secciones 2.2.2.2 y 2.2.2.3, para producir un peso compuesto [63]. El ponderado TF-IDF asigna al término *i* un peso en el texto *j*, como se muestra en la Ecuación (2.9).

$$w_{ij} = TF_{ij} \times IDF_j \quad (2.9)$$

Donde TF_{ij} es la frecuencia del término j en el texto i e IDF_j es la frecuencia inversa del término j, como se presentó en las Ecuaciones (2.7) y (2.8), respectivamente [63]. Reemplazando se tiene la Ecuación (2.10).

$$w_{ij} = \frac{f_{ij}}{MáxFreq_i} \times \log \frac{N}{n_j} \quad (2.10)$$

De esta manera, TF-IDF asigna al término *j* un peso en el texto *i* que es:

- Más grande cuando *j* está presente muchas veces dentro de un pequeño número de textos.
- Pequeño cuando el término aparece pocas veces en un texto, o aparece en muchos textos (se toma como una señal de baja relevancia).
- Más pequeño cuando el término aparece en casi todos los documentos.

2.2.3 Medida de similitud: Similitud de Coseno

En la generación automática de resúmenes, la similitud medida basada en coseno se utiliza de forma estándar para medir la similitud entre las representaciones vectoriales de textos. Esta similitud evalúa el valor del coseno del ángulo comprendido entre dos vectores a comparar, considerando la propiedad del coseno de ser igual a uno cuando los vectores son idénticos y a cero cuando son diferentes [68]. Así, entre más pequeño es el ángulo, mayor será la similitud, donde el valor de uno indica que las oraciones son exactamente iguales. De esta manera, dados los dos textos representados por los vectores $\overrightarrow{texto_i}$ y $\overrightarrow{texto_j}$ (ver Figura 2.3), la medida de similitud de cosenos será calculada como se observa en la Ecuación (2.11).

$$sim_{cos}(\overrightarrow{texto_i}, \overrightarrow{texto_j}) = \frac{\sum_{k=1}^{n} w_{ik} w_{jk}}{\sqrt{\sum_{k=1}^{n} w_{ik}^2 * \sum_{k=1}^{n} w_{jk}^2}} \quad (2.11)$$

Donde n es el número total de términos en el texto, w_{ik} se refiere al peso del término k en la oración $\overrightarrow{texto_i}$ y w_{jk} es el peso del término k en la oración $\overrightarrow{texto_j}$.

2.3 ALGORITMO FSP

FSP es una metaheurística inspirada en el proceso que lleva a cabo un pescador tradicional para encontrar el mejor lugar de pesca [59]. Inicialmente el algoritmo FSP identifica los puntos de captura (soluciones candidatas iniciales) para toda el área de pesca (espacio de búsqueda), cada punto de captura está compuesto por un vector de posición, x_i (localizado en el espacio de búsqueda) y una memoria de la mejor solución encontrada en la vecindad del punto de captura p_i. En la Ecuación (2.12) se muestra el conjunto de vectores de posición de los puntos de captura.

$$X = [x_1, x_2, \dots, x_i, \dots, x_N], i = 1,2, \dots, N \quad (2.12)$$

Donde x_i es el *i*-ésimo punto de captura del conjunto X y N es el número de puntos de captura.

Luego, el pescador toma como referencia uno de los puntos de captura x_i para arrojar la red de pesca (lanzamiento de red, l) un número (L) determinado de veces. La red de pesca abarca un área de tamaño predefinido que contiene un número M de puntos de red (posibles nuevos puntos de captura). El lanzamiento de la red de pesca sobre el punto de captura x_i se describe mediante la Ecuación (2.13).

$$y = [y_{i,2}, y_{i,2}, \dots, y_{i,k}, \dots, y_{i,M}], i = 1,2, \dots, N, k = 1,2, \dots, M \quad (2.13)$$

Donde Y es el conjunto de vectores de posición de la red para el *i*-ésimo punto de captura, $y_{i,k}$ es el *k*-ésimo punto de red encontrado desde el *i*-ésimo punto de captura y M es el número de puntos de la red.

Para simular la aleatoriedad del lanzamiento real de una red de pesca, el algoritmo crea los puntos de red de acuerdo a la Ecuación (2.14).

$$y_{i,k} = x_i + A_k \quad (2.14)$$

Donde A_k es un vector N-dimensional compuesto de números aleatorios en el rango $[-c, c]$, c es un número real denominado coeficiente de amplitud que representa el tamaño del área donde podría caer la red de pesca.

El pescador evalúa los puntos de red con el objetivo de encontrar un mejor punto de referencia desde el cual arrojar la red, si lo consigue el punto encontrado reemplaza al punto de captura correspondiente al finalizar la iteración. Este procedimiento se llevará a cabo un número de veces para cada punto de captura (número de iteraciones, T). El pescador siempre tiene el mejor punto de captura encontrado (memoria de la mejor solución global, G_{best}).

La Figura 2.4 muestra la transición de los puntos de captura entre dos iteraciones. En la primera iteración, el punto de captura uno (PC-1) encontró una mejor solución en el lanzamiento uno (triángulos) y el PC-4 en el lanzamiento tres (cuadrados). Por lo tanto para la iteración 2, estos puntos de captura son movidos al punto de red con el mejor resultado. En cambio los puntos PC-2 y PC-3 permanecen estáticos para la iteración dos, debido a que no se encontró en la red una mejor solución. Este procedimiento se repite hasta completar el número de iteraciones definido.

El esquema del algoritmo FSP se presenta en la Figura 2.5, inicialmente se generan los *N* puntos de captura, cada punto creado ($Inicializar\ x_i$) es evaluado ($Evaluar\ x_i$) y se establece como la mejor solución local ($Instarciar\ p_i$). Por último se verifica si el punto de captura es mejor que la mejor solución global ($Actualizar\ G_{best}$) y se repite el procedimiento hasta generar los N puntos de captura. Luego, el algoritmo itera T veces, en cada iteración se realiza una explotación por medio de L lanzamientos sobre cada x_i (punto de captura), que consiste en generar M puntos de red ($y_{i,k}$) al efectuar una modificación en x_i ($Crear\ y_{i,k}$). El $y_{i,k}$ se evalúa ($Evaluar\ y_{i,k}$) para verificar si es mejor que p_i ($Actualizar\ p_i$). Una vez que un x_i es explotado, se verifica si p_i es mejor que el x_i ($Actualizar\ x_i$) y después si es actualizado, se verifica si es mejor que G_{best} ($Actualizar\ G_{best}$). Por último el algoritmo aplica una estrategia para actualizar el coeficiente de amplitud c.

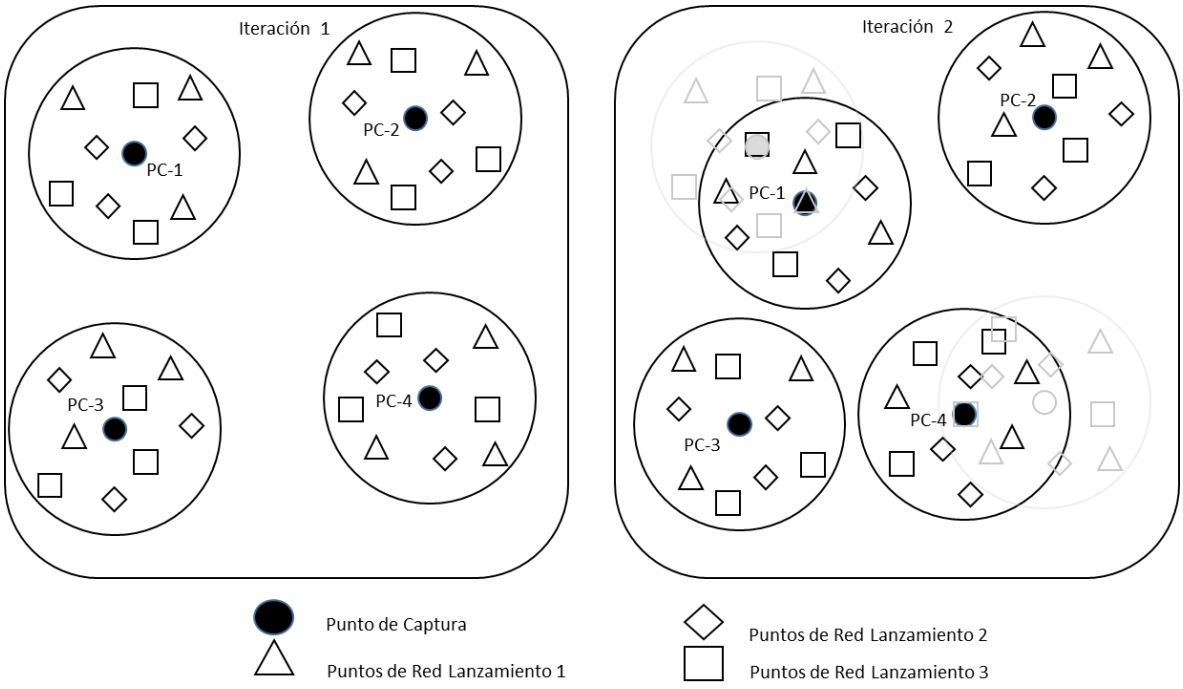

Figura 2.4 Grafico del FSP

```
INICIO FSP

HACER
    Inicializar x_i /*Crear el punto de captura i*/
    Evaluar x_i /*Evaluar la calidad del punto de captura i*/
    Instanciar p_i /*Establecer el punto de captura i como mejor solución local*/
    Actualizar G_best /*Actualizar la mejor solución global*/
HASTA Crear N Puntos de Captura

PARA t = 0 HASTA T HACER /*Numero de iteraciones*/
       PARA i = 0 HASTA N HACER /*Número de Puntos de Captura*/
              PARA l = 0 HASTA L HACER /*Numero de Lanzamientos*/
                     PARA k = 0 HASTA M HACER /*Número de Puntos de Red*/
                         Crear y_{i,k}  (y_{i,k} = x_i + A_k) /*Crear el punto de red k*/
                         Evaluar y_{i,k} /*Evaluar la calidad del punto red k*/
                         Actualizar p_i /*Actualizar la mejor solución local*/
                     FIN PARA
              FIN PARA
              Actualizar x_i /*Actualizar el punto de captura i*/
              Actualizar G_best /*Actualizar la mejor solución global*/
       FIN PARA
       Actualizar c /*Aplicar Estrategia de Actualización de c*/
FIN PARA
```

Figura 2.5. Esquema general FSP

2.4 ALGORITMOS DE BÚSQUEDA LOCAL

La búsqueda local es una heurística donde una solución inicial es modificada ligeramente aplicando operadores de variación que permiten buscar en su vecindad mejores soluciones. Si la solución encontrada es mejor, esta reemplaza a la solución inicial, de lo contrario se mantiene la solución inicial. El procedimiento se repite hasta encontrar un óptimo local (la mejor solución encontrada en la vecindad de la solución inicial) o hasta un número definido de iteraciones. La calidad del óptimo local obtenido depende de la solución inicial. En general se puede aplicar a diferentes problemas específicos de optimización combinatoria [73].

2.4.1 Ascenso de Colina

Es un algoritmo de búsqueda y optimización muy sencillo, que realiza un bucle en el que la mejor solución actual se utiliza para producir una nueva solución candidata. Si esta nueva solución es mejor que su predecesora, la sustituye y el ciclo comienza de nuevo [74]. Un número de iteraciones alto permite que el algoritmo aumente sus posibilidades de encontrar mejores soluciones, explotando una heurística de proximidad, donde pequeñas modificaciones a una solución candidata generalmente dan como resultado soluciones similares, que tiendan a comportarse de manera similar y a tener una calidad igual o superior, permitiendo paso a paso subir la colina hasta llegar a un óptimo local [73]. Algunas de las ventajas de este algoritmo son que requiere una cantidad limitada de memoria (almacena solo el estado actual) y es fácil de implementar.

En [73] se presenta un esquema del algoritmo Ascenso de la Colina, como se observa en la Figura 2.6. Donde una solución inicial (S) es copiada $(Copiar(S))$ y modificada $(Ajustar(Copiar(S)))$ para generar una nueva solución candidata (R), que es evaluada y comparada con S $(Calidad(R) > Calidad(S))$, de modo que si R es mejor que S, S se actualiza con la información de R, de lo contrario S se mantiene. El algoritmo itera hasta cumplir con una de las condiciones de salida que puede ser encontrar una solución S ideal, realizar un número máximo de evaluaciones o llevar a cabo un número fijo de iteraciones. Por último el algoritmo retorna la mejor solución encontrada $(Retornar\ S)$.

```
S ← Solución candidata inicial
Repetir
       R ← Ajustar (Copiar(S))
       Si Calidad(R) > Calidad(S) entonces
              S ← R
Hasta que S sea la solución ideal o se cumpla la condición de salida.
Retornar S.
```

Figura 2.6. Esquema general del algoritmo Ascenso de la Colina

2.4.2 Temple Simulado

Es un algoritmo de optimización propuesto por Kirkpatrick en 1983 [75] que puede ser aplicado a diversos problemas. Parte de una solución inicial y aplica un método para la generación de vecindario. Recibe su nombre del proceso de enfriamiento del metal, que

tiene como objetivo alterar mediante la variación de la temperatura sus propiedades. Si se deja enfriar el metal rápidamente los átomos se congelan en una configuración aleatoria, lo que resulta en metal quebradizo. En cambio sí se disminuye la temperatura muy lentamente, se da tiempo suficiente a los átomos para asentarse en una red fuerte [74].

Este algoritmo implementa un mecanismo que permite la elección de una solución de "mala calidad" en lugar de la solución de "mejor calidad". Si la nueva solución es mejor que la anterior, la reemplaza. De lo contrario hace el reemplazo por una solución de menor calidad con una cierta probabilidad, la cual disminuye durante la ejecución del algoritmo [76].

Para determinar esta probabilidad se utiliza un parámetro de temperatura (t). Cuando los valores de t son más altos, las soluciones de "mala calidad" son más aceptadas, a medida que el valor de t decrece, es menos probable aceptar soluciones de mala calidad, hasta que el algoritmo se comporta como el algoritmo Ascenso de la Colina (aceptando solo las mejores soluciones). Por lo tanto una de las características más importantes de este algoritmo es la elección del esquema de templado, el cual establece el ritmo al que se baja la temperatura, ya que si t disminuye lo suficientemente despacio, el algoritmo podrá encontrar un óptimo global [76].

En [73] se presenta un esquema del algoritmo Temple Simulado, como se observa en la Figura 2.7. Donde el parámetro de entrada temperatura (t) se establece en un valor inicial y se toma una solución (S) que se estable como mejor solución $(Best)$. El algoritmo comienza a iterar y en cada iteración S es copiada $(Copiar(S))$ y modificada $(Ajustar(Copiar(S)))$ para generar una nueva solución candidata (R), que es evaluada y comparada con S $(Calidad(R) > Calidad(S))$, de modo que si la calidad de R es mayor que la calidad de S o si el numero aleatorio ($Num_{Aleatorio} \in [0,1]$) es menor al factor $e^{\frac{\text{Calidad(R)} - \text{Calidad(S)}}{t}}$, S se actualiza con la información de R, de lo contrario S se mantiene. Luego se disminuye t y se verifica si la calidad de S es mayor a la calidad de $Best$, si esto se cumple $Best$ se actualiza con la información de S. Las iteraciones se realizan hasta cumplir con una de las condiciones de salida que puede ser encontrar una solución S ideal, realizar un número máximo de evaluaciones o llevar a cabo un número fijo de iteraciones. Por último el algoritmo retorna la mejor solución encontrada $(Retornar\ Best)$.

```
t ← Temperatura inicial
S ← Solución candidata inicial.
Best ← S, Se inicializa la mejor solución global.
Repetir
       R ← Ajustar (Copiar(S))
       Num_Aleatorio ← Aleatorio (0...1)
       Si Calidad(R) > Calidad(S) O
                                  Calidad(R) − Calidad(S)
                                        t
          Num_Aleatorio < e                    entonces
              S ← R
       Disminuir (t)
       Si Calidad(S) > Calidad (Best)
              Best ← S
Hasta que Best sea la solución ideal o se cumpla la condición de salida o t ≤ 0
Retornar Best.
```

Figura 2.7. Esquema general del algoritmo Temple Simulado

2.4.3 Búsqueda Tabú

Según Fred Glover en [77] "la búsqueda tabú es una metaheurística que guía un procedimiento heurístico de búsqueda local". Este algoritmo combina conceptos de inteligencia artificial y optimización al incorporar memoria adaptativa en la implementación de estructuras y procedimientos capaces de realizar un proceso de búsqueda eficaz y eficiente. Las estructuras de memoria tabú actúan siguiendo cuatro enfoques principales: (i) el enfoque en lo reciente y en la frecuencia para lograr el balance entre intensificación y diversificación; (ii) el enfoque en la calidad permite diferenciar la bondad de las soluciones visitadas a lo largo del procedimiento de resolución del problema, con el objetivo de reforzar las acciones que conducen a buenas soluciones y penalizar aquellas que conducen a malas soluciones; (iii) el enfoque basado en la flexibilidad de las estructuras de memoria que permite guiar la búsqueda en diferentes entornos; (iv) el enfoque en la influencia posibilita considerar la influencia de las decisiones tomadas durante la búsqueda, tanto para calidad de las soluciones, como para las estructuras de memoria que aprovechan la información obtenida a lo largo del procedimiento de resolución del problema y que permiten explotar tanto soluciones buenas y malas que aumentan las probabilidades de escapar de óptimos locales [78].

La memoria es uno de los componentes principales de la búsqueda tabú, la cual puede ser Explícita, atributiva o una combinación de ambas. La memoria Explícita guarda soluciones completas, que típicamente son soluciones ya visitadas durante la búsqueda. Por su parte, la memoria atributiva almacena información de la solución que puede ser los atributos (características) que componen a la solución o los movimientos de los atributos que se usaron para generarla [77].

El enfoque más sencillo para implementar una Búsqueda Tabú (ver Figura 2.8), es incluir como algoritmo de búsqueda local el Ascenso de la Colina y mantener una lista de soluciones candidatas que han sido probadas para que sean tenidas en cuenta por un tiempo, permitiendo que el proceso de optimización tenga menos posibilidades de quedarse atrapado en un óptimo local. Cuando la lista tabú llega a su longitud máxima se quita la solución candidata más antigua para agregar la nueva [73].

En el esquema presentado en la Figura 2.8, el algoritmo Búsqueda Tabú inicia estableciendo: la longitud de la lista tabú o tabú tenue (l), una solución inicial (S) que se toma como la mejor solución $(Best)$ y una lista tabú (L) con longitud (l), además se agrega S dentro de L. El algoritmo en cada iteración si la longitud de L es mayor que l, elimina el elemento más antiguo en la lista. A continuación S es copiada $(Copiar(S))$ y modificada $(Ajustar(Copiar(S)))$ para generar una nueva solución candidata (R). Luego se evalúa y compara la calidad de R con la calidad de S $(Calidad(R) > Calidad(S))$, de modo que si la calidad de R es mayor que la calidad de S y R no está en la lista L, S se actualiza con la información de R y R se agrega a L, de lo contrario S se mantiene. Por último se verifica si la calidad de S es mayor a la calidad de $Best$, si esto se cumple $Best$ se actualiza con la información de S. Las iteraciones se realizan hasta cumplir con una de las condiciones de salida que puede ser encontrar una solución S ideal, realizar un número máximo de evaluaciones o llevar a cabo un número fijo de iteraciones. Por último el algoritmo retorna la mejor solución encontrada $(Retornar\ Best)$.

```
l ← Tamaño máximo de la tabú (tenure)
S ← Solución candidata inicial.
Best ← S, Se inicializa la mejor solución global.
L ← Lista Tabú de longitud l                ► Lista tipo FIFO
Agregar S dentro de L
Repetir
       Si Longitud (L) > l entonces
              Eliminar último elemento en L
       R ← Ajustar (Copiar(S))
       Si R ∉ L Y Calidad(R) > Calidad(S) entonces
              S ← R
              Agregar R dentro de L
       Si Calidad(S) > Calidad (Best) entonces
              Best ← S
Hasta que Best sea la solución ideal o se cumpla la condición de salida.
Retornar Best.
```

Figura 2.8. Esquema general del algoritmo Búsqueda Tabú

Capítulo 3

3 PROCESO DE CONSTRUCCIÓN: ALGORITMO FSP-MT-SingleDocSum

Para el desarrollo de este proyecto se utilizó el Patrón de Investigación Iterativa (PII) propuesto por Pratt [60] diseñado especialmente para proyectos de investigación que involucran una solución computacional. Está compuesto por cuatro etapas principales que son: observación, identificación del problema, desarrollo de la solución y prueba de la solución. A continuación se realiza la descripción de los cuatro ciclos realizados. En el primero contemplando la definición de la función objetivo y la adaptación del algoritmo FSP original al problema de generación automática de resúmenes, con su correspondiente configuración y afinamiento de parámetros. Los ciclos dos, tres y cuatro, enfocados en la adaptación de los algoritmos FSP con ascenso de colina, FSP con temple simulado y FSP con una memoria Tabú, respectivamente, junto con la configuración y afinamiento de parámetros de cada algoritmo.

3.1 CICLO I: DISEÑO ALGORITMO FSP-SingleDocSum

En este ciclo se definen tres configuraciones de la función objetivo para medir la calidad de las soluciones generadas; la configuración de los parámetros del algoritmo FSP adaptados al problema de la generación automática de resúmenes y la configuración de los parámetros asociados al problema. También se realiza el afinamiento de los pesos de las tres configuraciones de la función objetivo para obtener la función definitiva que se utiliza en el algoritmo FSP-SingleDocSum. Finalmente se realiza el afinamiento de los parámetros del FSP y los asociados al problema.

3.1.1 Diseño de la función objetivo

La función objetivo es uno de los principales mecanismos que aproximan conocimiento del problema específico, ésta rige la exploración de soluciones evaluando su competencia para resolver el problema abordado y determinar su calidad. Para realizar el diseño de la función objetivo se determinó usar como configuración base el conjunto de características utilizadas en [51, 79, 80] que permiten evaluar la calidad de las oraciones de una solución candidata (resumen) de forma independiente al lenguaje y que mostraron un buen desempeño al mejorar los resultados de las medidas ROUGE sobre los resúmenes generados. A continuación se describe las características.

3.1.1.1 Posición de la oración en el documento

De acuerdo a [81] la información relevante suele ser encontrada en algunas secciones como títulos, encabezados, primeras oraciones de los párrafos, etc. Diferentes esquemas que evalúan las oraciones respecto a su posición en el documento han sido aplicados [49, 51, 82] mostrando su efectividad para determinar la relevancia de una oración.

- Posición 1

Este esquema propuesto por Bossard [83], aplica un cálculo del factor basado en la posición, que usa la distancia existente entre la oración y el inicio del documento. El puntaje más alto es asignado a la primera oración y disminuye a medida que las oraciones se alejan del principio del documento. La disminución de los puntajes es mayor al principio, por lo que, la diferencia entre los puntajes de las oraciones iniciales es mayor a la de las oraciones en posiciones medias y bajas, donde la disminución es gradual. El cálculo de esta característica está definido como se observa en la Ecuación (3.1), de modo que el puntaje del resumen candidato es la sumatoria de los puntajes de cada una de las oraciones que lo componen.

$$P(s_i) = \sum_{\forall s_i \in S} \sqrt[2]{\frac{1}{n_i}} \quad (3.1)$$

Donde n_i indica la posición de la oración s_i en el documento.

- Posición 2

En [79], se define un esquema que también utiliza la distancia existente entre la oración y el inicio del documento, sin embargo, el puntaje asignado va disminuyendo gradualmente desde la primera oración hasta la última, de acuerdo a la Ecuación (3.2). Esta característica tiende a $\frac{2}{O}$ cuando las oraciones en el resumen candidato corresponden a las primeras oraciones del documento y tiende a cero cuando las oraciones están en las últimas posiciones.

$$PF_S = \frac{\sum_{\forall s_i \in S} PositionRanking(s_i)}{O} \quad (3.2)$$

Donde *O* es el número de oraciones en el resumen S y $PositionRanking(s_i)$ es la clasificación de la posición de cada oración s_i calculada por la Ecuación (3.3).

$$PositionRanking(s_i) = \frac{2 - 2 * \left(\frac{pos_i - 1}{n - 1}\right)}{n} \quad (3.3)$$

Donde pos_i es la posición de la oración i según el orden de aparición en el documento, y n es el número total de oraciones en el documento. Esta fórmula se basa en el método de selección lineal de rango usada en algoritmos genéticos. El mejor en la clasificación recibe un valor de $\frac{2}{n}$ y el más bajo en la clasificación un valor cercano a cero.

3.1.1.2 Longitud de la oración

Esta característica permite estimar de acuerdo a la longitud (medida en palabras) de una oración su relevancia para el documento. Algunos estudios concluyen que las oraciones cortas deberían tenerse menos en cuenta para aparecer en el resumen de un documento [18].

- Longitud 1

En [51] se aplica un mecanismo que utiliza la desviación estándar para obtener una evaluación balanceada de las oraciones en base a la longitud, que evalúa favorablemente los resúmenes compuestos de oraciones largas y oraciones de tamaño medio, de acuerdo a las fórmulas expresadas en las Ecuaciones (3.4) y (3.5).

$$L = \sum_{\forall s_i \in S} \frac{1 - e^{-\alpha_i}}{1 + e^{-\alpha_i}} \quad (3.4)$$

$$\alpha_i = \frac{l_i - \mu(l_i)}{std(l_i)} \quad (3.5)$$

Donde l_i es la longitud de la oración s_i, $\mu(l_i)$ es la longitud media de las oraciones y $std(l_i)$ es la desviación estándar de las longitudes de las oraciones.

- Longitud 2

Este esquema resulta de modificar la ecuación (3.4), para enfatizar la búsqueda de soluciones con menos oraciones cortas. Para lograr esto se calcula el promedio de las longitudes de las oraciones del resumen (dividendo entre el número de oraciones, *N*), de tal forma que cuando *N* es grande (oraciones cortas) se disminuye el valor de esta característica en la función objetivo, comparado con un resumen cuyo *N* es más pequeño (oraciones largas). La modificación de esta fórmula se muestra en la ecuación (3.6) y la ecuación (3.5) se mantiene igual.

$$L = \frac{\sum_{\forall s_i \in S} \frac{1 - e^{-\alpha_i}}{1 + e^{-\alpha_i}}}{N} \quad (3.6)$$

3.1.1.3 Relación con el Título

Mide la similitud de las oraciones del resumen con el título del documento asumiendo que las oraciones más importantes son similares a éste [84]. Como en [42, 45, 51] esta característica calcula la similitud de coseno entre los vectores que representan cada oración s_i que compone el resumen y el vector que representa el título t, usando las Ecuaciones (3.7) y (3.8)

$$TR_S = \frac{\sum_{s_i \in S} sim(s_i, t)}{L} \quad (3.7)$$

$$TRF_S = \frac{TR_s}{máximo_{\forall S} TR} \quad (3.8)$$

Donde TR_S es el promedio de la similitud de las oraciones con el título en el resumen S, L es el número de oraciones del resumen, TRF_S es el factor de similitud de las oraciones del resumen S con el título y $máximo_{\forall S} TR$ es el promedio de los máximos valores obtenidos de las similitudes de las oraciones con el título. Si TR_S es cercano a cero las oraciones del resumen son distintas al título y recíprocamente cuando el valor es cercano a 1.

3.1.1.4 Cohesión

Esta característica mide la relación entre las oraciones del resumen para establecer si tratan sobre la misma información, y de este modo obtener un resumen con oraciones estrechamente relacionadas.

- Cohesión 1

En [42, 45, 51] la evaluación de la cohesión de un resumen se efectúa midiendo la similitud de cosenos entre los vectores que representan las oraciones que lo componen, como se muestra en las ecuaciones (3.9)-(3.12)

$$CF = \frac{log(C_s * 9 + 1)}{log(M * 9 + 1)} \quad (3.9)$$

$$C_s = \frac{\sum_{\forall s_i, s_j \in S} sim(s_i, s_j)}{N_s} \quad (3.10)$$

$$N_s = \frac{(L) * (L-1)}{2} \quad (3.11)$$

$$M = máxima\ Sim(i,j), \qquad i,j \leq N \quad (3.12)$$

Donde CF corresponde al factor de cohesión de un resumen, C_s, como se observa en la Ecuación (3.10), es el promedio de similitud de todas las oraciones en el resumen S, $sim(s_i, s_j)$ es la similitud entre las oraciones s_i y s_j, L es el número de oraciones del resumen, N_s, como se presenta en la Ecuación (3.11), es la cantidad de relaciones de similitud diferentes de cero en el resumen, M corresponde a la máxima similitud de las oraciones y N es la cantidad de oraciones en el resumen S (Ver Ecuación (3.12)).

- Cohesión 2

Este esquema resulta de ajustar Cohesion 1. Esto debido a diferencias en la matriz de similitud de oraciones con [51], la cual aunque presenta valores en el rango entre cero y uno, no se encuentran distribuidos en este rango. En este proyecto se normaliza la matriz representando los valores de las similitudes entre las oraciones a través de todo el rango de valores, debido a esto las ecuaciones (3.9) y (3.12) que utilizan la función logaritmo para ajustar los valores de la ecuación (3.10) entre 0 y 1, no son necesarias. De esta forma la cohesión se calcula directamente con la ecuación (3.13).

$$CF = \frac{\sum_{\forall s_i, s_j \in S} sim(s_i, s_j)}{N_s} \quad (3.13)$$

Donde CF es el promedio de las similitudes de todas las oraciones en el resumen S, que corresponde a la cohesión del resumen S, $sim(s_i, s_j)$ es la similitud de cosenos entre las oraciones S_i y S_j, N_s es el número de comparaciones de similitud entre las oraciones del resumen, como se presenta en la Ecuación (3.11). De este modo CF tiende a cero cuando las oraciones del resumen son muy diferentes entre ellas, mientras que CF tiende a uno cuando las oraciones son muy similares entre ellas.

3.1.1.5 Cobertura

La cobertura intenta medir que proporción de la información del texto original es cubierta por las oraciones que componen el resumen [85].

- Cobertura 1

En [51], la cobertura ha sido definida como la similitud entre el vector de cada oración que compone el resumen y el vector de todas las oraciones del documento, de este modo la cobertura se calcula según la Ecuación (3.14).

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} [sim(\vec{D}, \vec{s_i}) + sim(\vec{D}, \vec{s_j})] \quad (3.14)$$

Donde n es el número de oraciones del documento, $\vec{D}$ es el vector de pesos de los términos del documento, $\vec{s_i}$ corresponde al vector de pesos de los términos de la oración s_i y $\vec{s_j}$ es el vector de pesos de los términos de la oración s_j.

- Cobertura 2

Por otro lado, en [79, 80] se presenta un esquema diferente para la evaluación de esta característica, calculando la similitud de cosenos entre el vector de términos que representa el resumen y el vector de términos del documento, como se muestra en la ecuación (3.15).

$$Cov = sim_{cos}(R, D) \quad (3.15)$$

Donde R representa el vector del resumen y D el vector de todo el documento. Cuando Cov tiene un valor cercano a uno significa que el resumen cubre una gran proporción del texto original y por lo tanto tiene una alta cobertura, mientras un valor cercano a cero indica que el resumen tiene baja cobertura.

3.1.2 Configuraciones de la Función Objetivo

Con las características de la función objetivo definidas, se procedió a configurar tres configuraciones de función objetivo y a evaluarlas para determinar cuál se adaptaba mejor a las necesidades del problema de la generación automática de resúmenes textuales.

3.1.2.1 Primera Función Objetivo

Esta configuración tomó el diseño de la función objetivo propuesta en [51] para medir la calidad de los resúmenes de texto generados, debido a los buenos resultados obtenidos en este estudio que lo ubicaron en primer lugar respecto a otros métodos del estado del arte. Esta función objetivo incluye las características: *Posición, Longitud, Relación con el título, Cohesión y Cobertura*. Por lo tanto, esta función objetivo se calcula de acuerdo a la Ecuación (3.16) y su correspondiente fórmula matemática se observa en (3.17).

$$f(x) = Posición\ 1 + Longitud\ 1 + Relación\ con\ el\ título + Cohesión\ 1 + Cobertura\ 1 \tag{3.16}$$

$$f(x) = \sum_{i=1}^{m}\left[\sqrt[2]{\frac{1}{n_i}} + \frac{1 - e^{-\frac{l(s_i)-\mu(l)}{std(l)}}}{1 + e^{-\frac{l(s_i)-\mu(l)}{std(l)}}} + \frac{sim(s_i, t)}{L * \ máximo_{\forall\ resumen} TR}\right] + \frac{log(C * 9 + 1)}{log(M * 9 + 1)} + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}[sim(\vec{D}, \vec{s_i}) + sim(\vec{D}, \vec{s_j})\ ] \tag{3.17}$$

3.1.2.2 Segunda Función Objetivo

Para esta configuración se diseñó una función objetivo con menos características para reducir su complejidad, conformada por la Posición y la Cobertura debido a la importancia que muestran en diversos estudios del estado del arte [79, 80] junto a la cohesión utilizada en [51]. Por otro lado no se incluyeron las características de Relación con el Titulo y Longitud, considerando que la primera estaba incluida en la característica de cobertura y la segunda dado que en la mayoría de los estudios del estado del arte no es relevante. De esta forma, la función objetivo es calculada de acuerdo a la Ecuación (3.18), con su correspondiente fórmula matemática en (3.19).

$$f(x) = Posición\ 2 + Cohesión\ 2 + Cobertura\ 2 \tag{3.18}$$

$$f(x) = \sum_{\forall s_i \in Summary}\left[\frac{PositionsRanking(s_i)}{O} + \sum_{\forall s_j \in Summary}\frac{sim_{cos}(s_i, s_j)}{N_s}\right] + sim_{cos}(R, D) \tag{3.19}$$

3.1.2.3 Tercera Función Objetivo

Para esta configuración se utilizó las cinco características de la primera función objetivo, cambiando la forma de calcular de todas ellas (exceptuando la Relación con el Titulo), de la siguiente manera: la Posición, Cohesión y Cobertura se tomaron de la segunda función objetivo debido a los buenos resultados obtenidos (ver Tabla 6), la Longitud con las modificaciones mencionadas en la sección 3.1.1 y la Relación con el Titulo. De este modo los valores de las características de la función objetivo quedan en un rango entre 0 y 1; y es calculada como se presenta en la Ecuación (3.20), con su correspondiente fórmula matemática en (3.21).

$$f(x) = Posición\ 2 + Longitud\ 2 + Relación\ con\ el\ título + Cohesión\ 2 + Cobertura\ 2 \tag{3.20}$$

$$f(x) = \sum_{\forall s_i \in Summary} \left[\frac{PositionsRanking(s_i)}{0} + \frac{\frac{1 - e^{-\frac{l(s_i)-\mu(l)}{std(l)}}}{1 + e^{-\frac{l(s_i)-\mu(l)}{std(l)}}}}{N} + \frac{sim(s_i, t)}{máximo_{\forall\ summary} TR} + \sum_{\forall s_j \in Summary} \frac{sim_{cos}(s_i, s_j)}{N_s} \right] + sim_{cos}(R, D) \tag{3.21}$$

3.1.3 Configuración parámetros algoritmo FSP

Para definir de manera preliminar los parámetros del algoritmo FSP, se tomó como referencia la evaluación y análisis de cada parámetro para un problema continuo definida en [59], ajustándola al problema de generación automática de resúmenes, el cual es discreto.

El algoritmo utiliza cinco parámetros independientes: el número de punto de captura, el número de iteraciones, el número de lanzamientos, el número de puntos de red y el coeficiente de amplitud.

- Número de Puntos de Captura (N): es un factor importante para lograr diversidad en el algoritmo ya que establece los puntos de exploración del espacio de búsqueda. Aunque se recomienda valores en el rango entre 80 y 100, no son viables para realizar la experimentación de este problema, debido a la restricción del número de evaluaciones de la función objetivo y a que la extensión del espacio de búsqueda de este problema es más pequeña en comparación a la de un problema continuo.

- Iteraciones (T) y Lanzamientos (L): el número de lanzamientos e iteraciones controla la extensión de búsqueda, debido a esto se recomienda usar un número pequeño de iteraciones y lanzamientos que se traduce en una búsqueda rápida y además permite al algoritmo estabilizarse y encontrar una respuesta dentro de los límites esperados.

- Red de pesca (M, representa el tamaño de la red de pesca): el ajuste de la cantidad de puntos de red de pesca, constituye un factor fundamental para lograr la eficiencia del algoritmo, de esta manera, la definición de un valor adecuado, debe tener en cuenta que: una gran red de pesca (aquella compuesta por muchos puntos de red), garantiza una exploración profunda, pero incrementa considerablemente la carga computacional porque tiene que hacer más evaluaciones de la función objetivo. Por otro lado, si la red de pesca es pequeña, la convergencia rápida del algoritmo no garantiza una buena solución para el problema.

- Coeficiente de amplitud (C): para establecer una relación efectiva con este parámetro es muy importante conocer el tipo de función y el problema que se está abordando. El coeficiente de amplitud está configurado por defecto para la explotación de un espacio continuo, para su adaptación al problema de generación automática de resúmenes (espacio binario), el valor asignado a éste parámetro indica el número de oraciones a modificar en la solución.

Para establecer la configuración preliminar de estos parámetros se realizó un proceso de afinación junto a la primera función objetivo con la combinación final de pesos presentada en [51], que se muestra en la ecuación (3.22).

$$f(x) = 0{,}35 * \text{Posición} + 0{,}29 * \text{Longitud} + 0{,}005 * \text{Cohesión} + 0{,}005 * \text{Cobertura} + 0{,}35 * \text{Relación con el título} \quad (3.22)$$

Por lo tanto la afinación preliminar obtuvo una combinación de valores para estos parámetros que se listan en la Tabla 1, junto a su abreviatura.

Parámetro	Valor
T	5
N	7
L	3
M	15
C	1

Tabla 1. Configuración preliminar de los parámetros del FSP

3.1.4 Configuración parámetros asociados al problema

En cuanto a los parámetros asociados al problema se tomaron como base los presentados en [51], teniendo en cuenta que se trata del mismo problema. Estos parámetros son los siguientes:

- Máximo Número de Evaluaciones de la función objetivo: permite que durante la ejecución del algoritmo FSP no se exceda el límite de evaluaciones de la función objetivo. Para realizar una comparación en condiciones similares con otros métodos metaheurísticos del estado del arte, el valor de este parámetro se estableció en 1600.

- Límite de cantidad de palabras del resumen: establece la longitud máxima medida en palabras que debe tener el resumen generado. El valor de este parámetro se estableció en 100 palabras para permitir la evaluación de los resúmenes generados mediante las medidas ROUGE sobre los conjuntos de datos DUC2001 Y DUC2002.

- Máxima longitud a evaluar de un resumen: permite que se exceda la cantidad de palabras de cada solución candidata, con el objetivo de realizar una mayor exploración del espacio de búsqueda al permitir una mayor cantidad de combinaciones de oraciones para conformar los resúmenes candidatos. Sin embargo, al terminar la ejecución del algoritmo se valida que la cantidad de palabras que conforman el resumen no supere límite de cantidad de palabras del resumen.

- Criterio para deshabilitar una oración: establece que característica(s) de la función objetivo serán utilizadas para calcular el puntaje de cada oración del documento, y seleccionar la oración con el puntaje más bajo para ser deshabilitada. Se consideró de manera preliminar que la cobertura era la característica más prometedora para medir este criterio, ya que valora cada oración por su similitud al documento.

- Criterio de selección de oraciones del resumen final: define que característica(s) de la función objetivo serán utilizadas para especificar el orden en que las oraciones aparecen en el resumen generado. De manera preliminar se definió seleccionar la Posición para medir este criterio, debido al buen desempeño mostrado en [51] al utilizarla para esta tarea.

La configuración preliminar de estos parámetros se lista en la Tabla 2.

Parámetro	Valor
Máximo Número de Evaluaciones de la función objetivo	1600
Máxima longitud a evaluar de un resumen	100
Criterio para deshabilitar una oración	Cobertura
Criterio de selección de oraciones del resumen final	Posición

Tabla 2. Configuración preliminar de los parámetros asociados al problema.

3.1.5 Afinación de las funciones objetivo

Para dar flexibilidad a la función objetivo, a cada característica se le asocio un peso, que permite asignar un mayor o menor valor a cada una de ellas de acuerdo al desempeño de la función objetivo en el problema. Esta asignación de pesos se realiza con el objetivo de lograr que el impacto sobre el valor de la función objetivo de los valores obtenidos por las características sea equilibrado (ya que estos no se encuentran en el mismo rango) y de esta forma evitar que las características que obtienen valores muy grandes afecten de manera desproporcionada la evaluación de la función objetivo o por el contrario si son muy pequeños no la afecten. Para realizar el afinamiento de estos pesos en las tres configuraciones de las funciones objetivos, se tomó la configuración preliminar de los parámetros del algoritmo FSP (véase Tabla 1) y los parámetros asociados al problema (véase Tabla 2).

El afinamiento de estos pesos asociados a las características se realizó conformando grupos de experimentos, donde cada grupo define un conjunto de combinaciones de pesos, de acuerdo al rango de valores que puede tomar cada peso y al valor de incremento desde el valor mínimo hasta el máximo. Para mayor agilidad en el proceso de afinamiento se definió como restricción que la suma de estos pesos debía ser igual en cualquiera de las combinaciones establecidas.

La configuración del grupo inicial de los pesos de todas las características de la función objetivo se estableció en un rango entre 0 y 1, y un incremento de 0.2. Los mejores resultados de éste grupo, se seleccionan para determinar los nuevos rangos del siguiente grupo teniendo en cuenta que el incremento disminuye de acuerdo al rango. Este

procedimiento se aplica nuevamente hasta máximo cinco grupos dependiendo de las características de cada función objetivo y la variabilidad de los pesos.

3.1.5.1 Afinación Primera Función Objetivo

La configuración de la primera función objetivo está definida como se indica en la Ecuación (3.16), con la disposición de los pesos asociados a cada característica, de acuerdo a la Ecuación (3.23).

$$f(x) = \alpha * \text{Posición } 1 + \beta * \text{Longitud } 1 + \gamma * \text{Cohesión } 1 + \mu * \text{Cobertura } 1 + \rho * \text{Relación con el título} \quad (3.23)$$

Después de ejecutar el proceso de afinación, la combinación de pesos que obtuvo mejores resultados se muestra la Tabla 3.

Posición	Longitud	Cobertura	Cohesión	Relación con el título
α	β	γ	μ	ρ
0,05	0,40	0,05	0,20	0,30

Tabla 3. Pesos de la Primera Función Objetivo

3.1.5.2 Afinación Segunda Función Objetivo

La configuración de la segunda función objetivo está definida de acuerdo a la Ecuación (3.18), y con los pesos asociados a cada característica como se muestra en la Ecuación (3.24).

$$f(x) = \alpha * \text{Cobertura } 2 + \beta * \text{Cohesión } 2 + \gamma * \text{Posición } 2 \quad (3.24)$$

Después de ejecutar el proceso de afinación, la combinación de pesos que obtuvo mejores resultados se muestra en la Tabla 4.

Cobertura	Cohesión	Posición
α	β	γ
0,73	0,02	0,25

Tabla 4. Pesos de la Segunda Función Objetivo.

3.1.5.3 Afinación Tercera Función Objetivo

La configuración de la tercera función objetivo está definida como se indica en la Ecuación (3.20), con la disposición de los pesos asociados a cada característica, de acuerdo a la Ecuación (3.25).

$$f(x) = \alpha * \text{Posición } 2 + \beta * \text{Longitud } 2 + \gamma * \text{Relación con el título} + \mu * \text{Cohesión } 2 + \rho * \text{Cobertura } 2 \quad (3.25)$$

Después de ejecutar el proceso de afinación, la combinación de pesos que obtuvo mejores resultados se muestra Tabla 5.

Posición	Relación con el título	Longitud	Cohesión	Cobertura
α	β	γ	μ	ρ
0,15	0,05	0,1	0,05	0,65

Tabla 5. Pesos de la Tercera Función Objetivo.

3.1.6 Configuración Definitiva de la Función Objetivo

Los resultados de la evaluación de las tres funciones objetivo afinadas se presentan en la Tabla 6, donde se observa que la tercera función obtiene los mejores resultados en todas las medidas ROUGE sobre DUC2001 y los segundos mejores resultados sobre DUC2002. Además considerando que esta configuración al estar conformada por más características tiene mayores posibilidades de mejora respecto a la segunda función. De esta manera se decide establecer la tercera función como la función objetivo definitiva, con sus correspondientes pesos como se presenta en la Ecuación (3.26).

$$f(x) = 0.15 * \text{Posición2} + 0.05 * \text{Relación con el título} + 0.1 * \text{Longitud2} + 0.05 * \text{Cohesión} + 0.65 * \text{Cobertura2} \quad (3.26)$$

Función Objetivo	DUC2001			DUC2002		
	Rouge 1 – Recall	Rouge 2 - Recall	Rouge SU4 - Recall	Rouge 1 - Recall	Rouge 2 - Recall	Rouge SU4 - Recall
Primera Función Objetivo	0,38619	0,15090	0,17650	0,41370	0,17107	0,19355
Segunda Función Objetivo	0,44214	0,19598	0,21796	**0,47062**	**0,22051**	**0,23651**
Tercera Función Objetivo	**0,44636**	**0,20231**	**0,22289**	0,46834	0,21744	0,23396

Tabla 6. Resultados de afinación de las funciones objetivo

En esta configuración final se observa la importancia de características como la *Posición, y Relación con el título* al igual que en otras investigaciones previas del estado del arte [42, 45, 70, 86]. También de la Cobertura presente en estudios más recientes que han mostrado resultado prometedores [51, 79].

3.1.7 Afinación de parámetros del FSP

Partiendo de la configuración final de la función objetivo, el proceso de afinación de los parámetros del FSP se realizó en tres pasos:

- Primer paso. Conformación de un conjunto de combinaciones de valores para los parámetros (T, N, L y M), que debían cumplir con las siguientes restricciones: todos los parámetros pueden tomar valores en el rango de uno a treinta, el producto de estos parámetros debe estar en el rango de mil quinientos cincuenta mil seiscientos (para generar combinaciones que cumplan el límite de evaluaciones de la función objetivo), como se muestra en la ecuación (3.27). Una vez generado este conjunto, se descartan aquellas combinaciones que no presentan un buen balance entre exploración y explotación de acuerdo a la sección 3.1.3 (Configuración de parámetros algoritmo FSP).

$$1550 \leq (T * N * L * M) + N \leq 1600 \quad (3.27)$$

- Segundo paso. Se determinaron dos esquemas para el uso del parámetro C. En el primero el parámetro es estático, es decir, su valor no varía durante la ejecución del algoritmo, tomando valores pequeños (1 y 2), de esta forma al aplicar el ajuste no se modificaba demasiado las soluciones (evitando una exploración). En el segundo se

especifica un C dinámico, que varía durante la ejecución del algoritmo con el siguiente mecanismo: si el valor de C es uno se incrementa a dos y si el valor de C es dos se disminuye a uno, este cambio ocurre cuando el algoritmo no mejora durante dos iteraciones seguidas.

- Tercer paso. Evaluación del conjunto definido en el primer paso junto a los esquemas del segundo paso, para luego, analizar los resultados obtenidos y determinar que combinación de parámetros y esquemas obtiene el mejor desempeño. En la Tabla 7 se presenta el mejor resultado para cada esquema de C (estático y dinámico) luego del afinamiento de parámetros del FSP, donde se observa que el C estático muestra un mejor desempeño respecto al C dinámico.

Experimento	DUC2001			DUC2002		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
C Estático	**0,44607**	**0,20165**	**0,22251**	**0,47004**	**0,21838**	**0,23498**
C Dinámico	0,44607	0,20162	0,22248	0,47004	0,21838	0,23498

Tabla 7. Mejor Resultado obtenido para cada configuración del parámetro C

En la Tabla 8 se observa el valor de cada uno de los parámetros FSP con los que se obtuvo el mejor resultado presentado en la Tabla 7 (esquema C estático).

Parámetro	Valor
T	5
N	15
L	3
M	7
C Estático	1

Tabla 8. Mejor Combinación de los parámetros del FSP

3.1.8 Afinación de parámetros asociados al Problema

El proceso de afinación de cada uno de los parámetros asociados al problema se realizó, de la siguiente forma:

- Máxima Longitud a Evaluar de un resumen, tomando el valor de 100, 110, 120 y 130. Los mejores resultados se obtuvieron con el valor de 130, como se observa en la Tabla 9.

Experimento		DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
C-Estático	100	0,44607	0,20165	0,22251	0,47004	0,21838	0,23498
	110	0,45468	0,20221	0,22503	0,48433	0,22645	0,24304
	120	0,45846	0,20537	0,22756	0,48698	0,22868	0,24498
	130	**0,45946**	**0,20580**	**0,22827**	**0,48735**	**0,22935**	**0,24599**
C-Dinámico	100	0,44607	0,20162	0,22248	0,47004	0,21838	0,23498
	110	0,45400	0,20201	0,22479	0,48413	0,22645	0,24300
	120	0,45785	0,20460	0,22689	0,48714	0,22911	0,24528
	130	0,45944	0,20580	0,22826	0,48731	0,22932	0,24596

Tabla 9. Mejores resultados Máxima Longitud a Evaluar

Al finalizar la afinación de este parámetro, los valores de los parámetros del FSP correspondientes al experimento que obtuvo mejores resultados de acuerdo a los presentados en la Tabla 9, se muestran en la Tabla 10.

Parámetro	Valor
T	6
N	13
L	2
M	10
C Estático	1

Tabla 10. Mejor Combinación de los parámetros del FSP

- Criterio para Deshabilitar una oración, se seleccionaron individualmente las características de: Posición, Relación con el Titulo, Longitud, Cohesión y Cobertura, y las combinaciones de las características que mostraron mejores resultados. Los resultados obtenidos se presentan en la Tabla 11, en la cual se observa que la combinación de las características de Posición y Cobertura, presenta los mejores resultados.

- Criterio de Selección de Oraciones al generar el resumen, se experimentó con las características individuales y parejas que mejores resultados obtuvieron, de acuerdo a los presentados en la Tabla 11, además el Criterio para Deshabilitar una oración se estableció en Posición-Cobertura debido a que mostro el mejor desempeño. Los resultados obtenidos al finalizar el proceso de afinación del parámetro *Criterio de selección de oraciones del resumen final*, se presentan en la Tabla 12, en la cual se observa que la característica con mejor desempeño fue la de Posición.

Experimento		DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
C-Estático	**Posición, Cobertura**	**0,45982**	**0,20589**	**0,22838**	**0,48809**	**0,23015**	**0,24666**
	Cobertura	0,45946	0,2058	0,22831	0,48735	0,22935	0,24599
	Posición	0,45825	0,20539	0,22825	0,48675	0,22907	0,24552
	Cohesión	0,45044	0,19644	0,22832	0,47955	0,22237	0,23994
	Similitud al Titulo	0,46040	0,20592	0,22876	0,48599	0,22796	0,24459
	Longitud	0,45899	0,20559	0,22840	0,48533	0,22782	0,24446
	Cobertura -Similitud al Titulo	0,46017	0,20641	0,22889	0,48761	0,22959	0,24610
C-Dinámico	Posición Cobertura	0,45975	0,20584	0,22847	0,48806	0,23014	0,24665
	Cobertura	0,45944	0,20580	0,22826	0,48731	0,22932	0,24596
	Posición	0,45827	0,20540	0,22771	0,48676	0,22910	0,24555
	Cohesión	0,44950	0,19573	0,21973	0,47896	0,22064	0,23861
	Similitud al Titulo	0,46043	0,20602	0,22883	0,48595	0,22792	0,24456
	Longitud	0,45895	0,20556	0,22837	0,48522	0,22768	0,24434
	Cobertura -Similitud al Titulo	0,46017	0,20644	0,22892	0,48760	0,22959	0,24610

Tabla 11. Mejor resultado Criterio para Deshabilitar una oración

Experimento		DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
C-Estático	**Posición**	**0,45982**	**0,20589**	**0,22852**	**0,48809**	**0,23015**	**0,24666**
	Cobertura	0,45568	0,20013	0,22367	0,47989	0,22160	0,23830
	Longitud	0,45347	0,19776	0,22126	0,47787	0,21975	0,23698
	Similitud al Titulo	0,45808	0,20187	0,22487	0,48039	0,22040	0,23739
	Cohesión	0,45818	0,20131	0,22528	0,48197	0,22252	0,23922
	Posición-Cobertura	0,45683	0,20235	0,22553	0,48324	0,22509	0,24148
C-Dinámico	Posición	0,45975	0,20584	0,22847	0,48806	0,23014	0,24665
	Cobertura	0,45564	0,20009	0,22364	0,47988	0,22160	0,23829
	Longitud	0,45344	0,19774	0,22124	0,47788	0,21977	0,23699
	Similitud al Titulo	0,45805	0,20183	0,22483	0,48035	0,22036	0,23735
	Cohesión	0,45816	0,20129	0,22526	0,48195	0,22251	0,23920
	Posición-Cobertura	0,45680	0,20233	0,22552	0,48324	0,22509	0,24147

Tabla 12. Mejor resultado Criterio de selección de oraciones del resumen final

Cabe destacar que en el proceso de afinación de los parámetros *Criterio para Deshabilitar una oración* y *Criterio de Selección de Oraciones al generar el resumen*, fue necesario una nueva afinación de los pesos de la función objetivo definitiva, ya que se utilizan algunas de sus características en estos parámetros. Al finalizar este proceso se encontró que los pesos obtenidos coinciden con los presentados en configuración definitiva de la función objetivo Ecuación (3.22).

Concluida la afinación de parámetros asociados al problema, los valores de los parámetros corresponden al experimento que obtuvo el mejor resultado se presentan en la Tabla 13.

Parámetro	Valor
Máximo Número de Evaluaciones de la función objetivo	1600
Máxima longitud a evaluar de un resumen	130
Criterio para deshabilitar una oración	Posición y Cobertura
Criterio de Selección de Oraciones al generar el resumen	Posición

Tabla 13. Configuración final de los parámetros asociados al problema.

3.1.9 Esquema Algoritmo FSP-SingleDocSum

El esquema del algoritmo FSP-SingleDocSum presentado en la Figura 3.1 es la adaptación del algoritmo FSP al problema de la generación automática de resúmenes de textos. Los principales cambios del algoritmo propuesto en este ciclo respecto al algoritmo FSP (ver Figura 2.5) son los siguientes: se estableció la representación de la soluciones binaria (ver sección 4.1); el parámetro C se estableció en 1 e indica el número de oraciones a modificar en el punto de captura (ver sección 3.1.3), además al ser estático no es necesario aplicar la estrategia de actualización de C; y el procedimiento para crear un punto de red se lleva a cabo en $Generar\ Vecino()$.

El procedimiento $Generar\ Vecino()$ presentado en la Figura 3.2 a diferencia del algoritmo FSP efectúa algunas tareas adicionales para crear un punto de red, que se realizan de la siguiente forma: inicialmente se genera una solución copia (aux_i) del punto de captura a ajustar ($Copiar(x_i)$), luego se evalúan las oraciones activas en la solución copia de acuerdo a las características establecidas en el *Criterio para Deshabilitar una oración,* y de este modo se selecciona la oración con la menor evaluación para ser deshabilitada $(Deshabilitar\ Oración\ en\ (aux_i))$. A continuación se habilita aleatoriamente una oración en la solución copia $(Habilitar\ Oración\ en\ (aux_i))$ y por último se realiza una comprobación y ajuste para que cumpla con el parámetro asociado al problema *Máxima longitud a Evaluar de un Resumen* $(Reparar\ Solucion\ (aux_i))$.

Una vez creado el punto de red se evalúa su calidad mediante la función objetivo definida en la sección 3.1.6 y a continuación se compara $(Actualizar\ p_i)$ con la mejor solución local (p_i), de este modo, si el punto de red es mejor, p_i se actualiza con su información. De lo contrario p_i se mantiene. Estas tareas se repiten hasta generar M puntos de red.

Cabe destacar que al retornar la mejor solución global G_{best} el algoritmo FSP-SingleDocSum ordena las oraciones según el *Criterio de Selección de Oraciones al generar el Resumen* para que aparezcan en ese orden en el resumen generado.

```
INICIO FSP
C=1 /*Se estable el coeficiente de amplitud*/
HACER
    Inicializar x_i /*Crear el punto de captura i*/
    Evaluar x_i /*Evaluar la calidad del punto de captura i*/
    Instanciar p_i /*Establecer el punto de captura i como mejor solución local*/
    Actualizar G_best /*Actualizar la mejor solución global*/
HASTA Crear N Puntos de Captura

PARA t = 0 HASTA T HACER /*Numero de iteraciones*/
       PARA i = 0 HASTA N HACER /*Número de Puntos de Captura*/
              PARA l = 0 HASTA L HACER /*Numero de Lanzamientos*/
                  PARA k = 0 HASTA M HACER /*Número de Puntos de Red*/
                     y_k = Generar Vecino(x_i)    /*Generar Punto de Red*/
                     Evaluar y_k /*Evaluar la calidad del punto red k*/
                     Actualizar p_i /*Actualizar la mejor solución local*/
              FIN PARA
              Actualizar x_i /*Actualizar el punto de captura i*/
              Actualizar G_best /*Actualizar la mejor solución global*/
       FIN PARA
FIN PARA
RETORNAR G_best
```

Figura 3.1 Esquema Algoritmo FSP-SingleDocSum

```
INICIO Generar Vecino

    aux_i = Copiar(x_i) /*Copiar el punto de captura i/*
    Puntuar Oraciones de (aux_i)
    Deshabilitar Oración en (aux_i)
    Habilitar Oración en (aux_i)
    Reparar Solución (aux_i)

RETORNAR aux_i

FIN Generar Vecinos
```

Figura 3.2 Esquema Procedimiento Generar Vecinos

3.2 CICLO II: DISEÑO ALGORITMO FSP CON ASCENSO DE LA COLINA

En este ciclo se define la configuración de los parámetros y tres esquemas de optimización del algoritmo Ascenso de la Colina para su adaptación al algoritmo FSP-

SingleDocSum. También se realiza el afinamiento de los parámetros del Ascenso de la Colina y FSP para cada uno de los esquemas. Finalmente se adapta al algoritmo FSP-SingleDocSum el esquema que obtuvo mejores resultados que determina la configuración definitiva del algoritmo para este ciclo FSP-ASC-SingleDocSum.

3.2.1 Configuración parámetros algoritmo FSP con Ascenso de la Colina

En este ciclo el algoritmo FSP incluye un optimizador de ascenso de colina, por lo tanto para su configuración es necesario agregar a los parámetros definidos en la sección 3.1.3, otros específicos concernientes a la optimización, que son:

- Numero de Optimizaciones (Opt). Determina el número de veces que el método de optimización local (definido para el ciclo) itera ajustando una solución, es decir, indica con que intensidad se realiza la búsqueda local sobre esa solución.

- Porcentaje de Puntos a optimizar (ProbabilidadOpt). Indica el porcentaje de puntos de captura que serán objeto de optimización.

3.2.2 Afinación de parámetros del FSP con Ascenso de la Colina

El proceso de afinación utiliza la configuración de la función objetivo definitiva (sección 3.1.6) y la configuración de los parámetros asociados al problema descrita en la Tabla 13. Este proceso comienza conformado tres conjuntos distintos de combinaciones de valores de los parámetros T, N, L, M, Opt y ProbabilidadOpt, que pueden tomar valores en el rango de uno a treinta a excepción de ProbabilidadOpt que varía entre 0.0 y 1.0. Cada conjunto de combinaciones se conformó para que se adaptara a uno de los siguientes esquemas de optimización:

- Optimización aleatoria después de la inicialización (OptAleatoriaInicializar). La optimización se realiza de forma aleatoria sobre un porcentaje de los puntos de captura iniciales. Las combinaciones del conjunto de experimentación para este esquema cumplen la restricción (3.28), que a diferencia de la Ecuación (3.27), suma la cantidad de ajustes realizados a cada uno de los puntos de captura a optimizar.

$$1550 \leq (T * N * L * M) + N + (N * ProbabilidadMejora * Opt) \leq 1600 \quad (3.28)$$

- Optimización aleatoria después de cada iteración (OptAleatoriaIterar). La optimización se aplica de forma aleatoria sobre un porcentaje de los puntos de captura al finalizar cada iteración. Las combinaciones del conjunto de experimentación para este esquema cumplen la restricción (3.29), que a diferencia de la Ecuación (3.27), agrega la cantidad de ajustes que se realizan sobre los $(N * ProbabilidadMejora)$ puntos de captura seleccionados para optimizar en cada iteración.

$$1550 \leq (T * N * L * M) + N + T * (N * ProbabilidadMejora * Opt) \leq 1600 \quad (3.29)$$

- Optimización del mejor después de cada iteración (OptMejorIterar). La optimización se aplica sobre el mejor punto de captura al finalizar cada iteración. Las combinaciones del conjunto de experimentación para este esquema cumplen la restricción (3.30), que

a diferencia de la Ecuación (3.27), añade la cantidad de ajustes realizados al punto de captura a optimizar de cada iteración.

$$1550 \leq (T * N * L * M) + N + (T * Opt) \leq 1600 \quad (3.30)$$

Una vez conformado el conjunto de combinaciones de cada esquema, se realiza su evaluación junto a los esquemas C (C-Estático y C-Dinámico), al igual que el proceso de afinación de los parámetros del FSP (ver 3.1.7).

El mejor resultado obtenido al finalizar el proceso de afinación para cada uno de los esquemas de optimización y el respectivo esquema C utilizado se muestran en la Tabla 14. De acuerdo a lo observado en esta tabla se establece que el esquema *OptAleatoriaAllInicializar* y *C Estático* obtiene el mejor desempeño al obtener los mejores resultados en cinco de las medidas (ROUGE 1 y ROUGE 2 sobre DUC2001 y ROUGE 1, ROUGE 2 y ROUGE SU4 sobre DUC2002) con los valores de los parámetros del FSP presentados en la Tabla 15.

Experimento		DUC2001			DUC2002		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
C – Estático	**OptAleatoriaAllInicializar**	**0,46027**	**0,20627**	0,22784	**0,48801**	**0,22992**	**0,24645**
	OptMejorAllterar	0,45956	0,20546	0,22853	0,48796	0,22977	0,24635
	OptAleatoriaAllterar	0,45967	0,20567	**0,22855**	0,4876	0,22954	0,24618
C – Dinámico	OptAleatoriaAllInicializar	0,45992	0,20569	0,22827	0,48794	0,22995	0,2465
	OptMejorAllterar	0,45956	0,20547	0,22808	0,48794	0,22975	0,24634
	OptAleatoriaAllterar	0,45977	0,20622	0,22813	0,48795	0,22999	0,24651

Tabla 14. Mejor Resultado esquemas de optimización y el parámetro C.

Parámetro	Valor
T	8
N	8
L	3
M	8
C Estático	1
Numero de Optimizaciones	5
Porcentaje de Puntos a Mejorar	0.2
Optimización	Aleatoriamente Luego de Inicializar

Tabla 15. Mejor configuración de parámetros FSP con Ascenso Colina

3.2.3 Esquema Algoritmo FSP-ASC-SingleDocSum

El esquema del algoritmo FSP-ASC-SingleDocSum presentado en la Figura 3.3 es la adaptación del algoritmo Ascenso de la Colina al algoritmo FSP-SingleDocSum. El algoritmo propuesto en este ciclo difiere respecto al FSP-SingleDocSum (ver Figura 3.1) en que agrega el procedimiento $Optimizar\ Puntos\ de\ Captura\ con\ Ascenso\ de\ la\ Colina()$.

```
INICIO FSP
C=1 /*Se estable el coeficiente de amplitud*/
HACER
    Inicializar x_i /*Crear el punto de captura i*/
    Evaluar x_i /*Evaluar la calidad del punto de captura i*/
    Instanciar p_i /*Establecer el punto de captura i como mejor solución local*/
    Actualizar G_best /*Actualizar la mejor solución global*/
HASTA Crear N Puntos de Captura
Optimizar Puntos de Captura con Ascenso de la Colina()
PARA t = 0 HASTA T HACER /*Numero de iteraciones*/
       PARA i = 0 HASTA N HACER /*Número de Puntos de Captura*/
              PARA l = 0 HASTA L HACER /*Numero de Lanzamientos*/
                  PARA k = 0 HASTA M HACER /*Número de Puntos de Red*/
                     y_k = Generar Vecino(x_i)   /*Generar Punto de Red*/
                     Evaluar y_k /*Evaluar la calidad del punto red k*/
                     Actualizar p_i /*Actualizar la mejor solución local*/
              FIN PARA
              Actualizar x_i /*Actualizar el punto de captura i*/
              Actualizar G_best /*Actualizar la mejor solución global*/
       FIN PARA
FIN PARA
RETORNAR G_best
```

Figura 3.3 Esquema Algoritmo FSP-ASC-SingleDocSum

El esquema de $Optimizar\ Puntos\ de\ Captura\ con\ Ascenso\ de\ la\ Colina()$ se presenta en la Figura 3.4. Este procedimiento adapta el algoritmo Ascenso de la Colina para optimizar los puntos de captura luego de que son creados, de la siguiente forma: establece dos parámetros de entrada Opt y $ProbabilidadOpt$, el primero indica el número de veces que un punto de captura es optimizado y el segundo indica el porcentaje de los N puntos de captura que serán optimizados. A continuación selecciona aleatoriamente uno de los puntos de captura a través de un número aleatorio $(NumeroAleatorio(1, N))$ y lo establece como solución inicial (S).

Para comenzar la optimización, S es copiada $(Copiar(S))$ y ajustada por el procedimiento $GenerarVecino()$ (ver Figura 3.2) para generar una nueva solución candidata R, que es evaluada con la función objetivo definida en la sección 3.1.6 y comparada con la evaluación de S, de este modo, si la evaluación de R es mayor que la evaluación de S, S se actualiza con R, de lo contrario S se mantiene. De esta manera S se optimiza Opt veces para al final actualizar el punto de captura (x_i). Este procedimiento se lleva a cabo en cada uno de los puntos de captura seleccionados aleatoriamente para ser optimizados.

```
INICIO Optimizar Puntos de Captura con Ascenso de Colina

REPETIR
        i = NumeroAleatorio(1, N)
        S = x_i  /*Solución Inicial*/
        PARA j = 0 HASTA Opt HACER /*Número de Optimizaciones */
                R = GenerarVecino(Copiar(S))
                Si Calidad(R) > Calidad(S) entonces
                        S ← R
        FIN PARA
        x_i = S
HASTA Optimizar (N*ProbabilidadOpt) Puntos de Captura

FIN Optimizar Puntos de Captura con Ascenso de Colina
```

Figura 3.4 Procedimiento de Optimización con Ascenso de la Colina

3.3 CICLO III: DISEÑO ALGORITMO FSP CON TEMPLE SIMULADO

En este ciclo se define la configuración de los parámetros y tres esquemas de optimización del algoritmo Temple Simulado para su adaptación al algoritmo FSP-SingleDocSum. También se realiza el afinamiento de los parámetros del Temple Simulado y FSP para cada uno de los esquemas. Finalmente se adapta al algoritmo FSP-SingleDocSum el esquema que obtuvo mejores resultados que determina la configuración definitiva del algoritmo para este ciclo FSP-TS-SingleDocSum.

3.3.1 Configuración parámetros algoritmo FSP con Temple Simulado

Los parámetros definidos para esta configuración, son los descritos en la sección 3.2.1 y el especificado por el algoritmo Temple Simulado: Temperatura (temp). Este parámetro ajustable, se establece inicialmente en un valor alto, para que el algoritmo tenga mayor probabilidad de aceptar soluciones de menor calidad y luego disminuye su valor poco a poco para lograr que esta probabilidad disminuya, usando el método de reducción lineal presentado en [87]. De esta forma se reduce gradualmente la temperatura inicial hasta cero, permitiendo con un número pequeño de iteraciones aumentar gradualmente la intensidad de la explotación del espacio de búsqueda al disminuir la aceptación de soluciones de menor calidad. Este algoritmo utiliza el número iteraciones para establecer tanto el valor inicial de la temperatura como el que tendrá en cada iteración, de acuerdo a la Ecuación (3.31).

$$temp(t) = (1 - \frac{t}{Opt})^{\propto} * temp_{Inicial} \quad (3.31)$$

Donde t indica el número de la iteración actual y puede tomar un valor entre [0, Opt] Opt es el número de iteraciones, $temp_{Inicial}$ es el valor de la temperatura inicial, toma el mismo valor de Opt, $\propto$ es una constante que establece el grado de la reducción polinómica. En este caso $\propto = 1$ para una reducción lineal.

3.3.2 Afinación de parámetros del FSP con Temple Simulado

La afinación utiliza la configuración final de función objetivo (sección 3.1.6) y la configuración de los parámetros asociados al problema presentados en la Tabla 13. Este proceso se realiza de igual forma a la afinación de parámetros del FSP (sección 3.2.2). El mejor resultado obtenido para cada uno de los esquemas de optimización y el respectivo esquema C utilizado, se presentan en la Tabla 16. De acuerdo a lo observado en esta tabla se establece que el esquema *OptMejorAlIterar* y *C Estático* obtiene el mejor desempeño al obtener los mejores resultados en todas las medidas ROUGE sobre DUC2001 y DUC2002 con los valores de los parámetros del FSP presentados en la Tabla 17.

Experimento		DUC2002			DUC2001		
		R1R	R2R	RSU4R	R1R	R2R	RSU4R
C – Estático	OptAleatoriaAlInicializar	0,45933	0,20562	0,22812	0,48788	0,22988	0,24635
	OptMejorAlIterar	**0,45970**	**0,20604**	**0,22791**	**0,48799**	**0,22992**	**0,24639**
	OptAleatoriaAlIterar	0,45967	0,20580	0,22846	0,48796	0,22990	0,24645
C – Dinámico	OptAleatoriaAlInicializar	0,45932	0,20562	0,22813	0,48786	0,22989	0,24636
	OptMejorAlIterar	0,45965	0,20600	0,22867	0,48797	0,22992	0,24639
	OptAleatoriaAlIterar	0,45961	0,20574	0,22772	0,48794	0,22989	0,24644

Tabla 16. Mejor Resultado esquemas de optimización y el parámetro C.

Parámetro	Valor
T	6
N	14
L	2
M	9
C Estático	1
Numero de Optimizaciones y valor temperatura	10
Porcentaje de Puntos a Mejorar	0.0
Optimización	Al Mejor Luego de Iterar

Tabla 17. Mejor Combinación de parámetros FSP con Temple Simulado

3.3.3 Esquema Algoritmo FSP-TS-SingleDocSum

El esquema del algoritmo FSP-TS-SingleDocSum presentado en la Figura 3.5 es la adaptación del algoritmo Temple Simulado al algoritmo FSP-SingleDocSum. El algoritmo propuesto en este ciclo difiere respecto al FSP-SingleDocSum (ver Figura 3.1) en que agrega el procedimiento $Optimizar\ el\ Mejor\ Punto\ de\ Captura\ con\ Temple\ Simulado()$.

```
INICIO FSP
C=1 /*Se estable el coeficiente de amplitud*/
HACER
    Inicializar x_i /*Crear el punto de captura i*/
    Evaluar x_i /*Evaluar la calidad del punto de captura i*/
    Instanciar p_i /*Establecer el punto de captura i como mejor solución local*/
    Actualizar G_best /*Actualizar la mejor solución global*/
HASTA Crear N Puntos de Captura
PARA t = 0 HASTA T HACER /*Numero de iteraciones*/
       PARA i = 0 HASTA N HACER /*Número de Puntos de Captura*/
              PARA l = 0 HASTA L HACER /*Numero de Lanzamientos*/
                  PARA k = 0 HASTA M HACER /*Número de Puntos de Red*/
                     y_k = Generar Vecino(x_i)    /*Generar Punto de Red*/
                     Evaluar y_k /*Evaluar la calidad del punto red k*/
                     Actualizar p_i /*Actualizar la mejor solución local*/
              FIN PARA
              Actualizar x_i /*Actualizar el punto de captura i*/
              Actualizar G_best /*Actualizar la mejor solución global*/
       FIN PARA
       Optimizar el Mejor Punto de Captura con Temple Simulado()
FIN PARA
RETORNAR G_best
```

Figura 3.5 Esquema Algoritmo FSP-TS-SingleDocSum

El esquema de $Optimizar\ el\ Mejor\ Punto\ de\ Captura\ con\ Temple\ Simulado()$ es presentado en la Figura 3.6. Este procedimiento adapta el algoritmo Temple Simulado para optimizar el mejor punto de captura, de la siguiente forma: establece dos parámetros de entrada Opt y t, el primero indica el número de veces que un punto de captura es optimizado y el segundo es la temperatura. A continuación selecciona el punto de captura con mejor evaluación y lo establece como solución inicial (S) y la mejor solución $(Best)$. Para comenzar la optimización, S es copiada $(Copiar(S))$ y ajustada por el procedimiento $GenerarVecino()$ (ver Figura 3.2) para generar una nueva solución candidata R, que es evaluada con la función objetivo definida en la sección 3.1.6 y comparada con la evaluación de S, de este modo, si la evaluación de R es mayor que la evaluación de S, S se actualiza con R o si el numero aleatorio ($Num_{Aleatorio} \in [0,1]$) es menor al factor $e^{\frac{\text{Calidad(R)} - \text{Calidad(S)}}{t}}$, S se actualiza con la información de R, de lo contrario S se mantiene. Luego se disminuye t de acuerdo a la Ecuación (3.31) y se verifica si la calidad de S es mayor a la calidad de $Best$, si esto se cumple $Best$ se actualiza con la información de S. De esta manera S se optimiza Opt veces para al final actualizar el punto de captura (x_i).

```
INICIO Optimizar Puntos de Captura con Ascenso de Colina
t = Opt
S = x_i  /*Solución Inicial*/
Best = S  /*Se inicializa la mejor solución global*/

PARA j = 0 HASTA Opt HACER /*Número de Optimizaciones */
       R = GenerarVecino(Copiar(S))
       Num_Aleatorio ← Aleatorio (0...1)
       SI Calidad(R) > Calidad(S) O
          Num_Aleatorio < e^((Calidad(R) − Calidad(S))/t)   entonces
             S = R
       FIN SI
       Disminuir (t)
       SI Calidad(S) > Calidad (Best)
             Best = S
       FIN SI
FIN PARA
x_i = Best
FIN Optimizar El mejor Punto de Captura con Temple Simulado
```

Figura 3.6 Procedimiento de Optimización con Temple Simulado.

3.4 CICLO IV: DISEÑO ALGORITMO FSP CON UNA MEMORIA TABU

En este ciclo se define la configuración de los parámetros de la memoria tabú para su adaptación al algoritmo FSP-SingleDocSum. También se realiza el afinamiento de los parámetros de la memoria tabú y FSP para cuatro tipos de memoria. Finalmente se adapta al algoritmo FSP-SingleDocSum el tipo de memoria tabú que obtuvo mejores resultados que determina la configuración definitiva del algoritmo para este ciclo FSP-MT-SingleDocSum.

3.4.1 Configuración de parámetros del algoritmo FSP

Los parámetros definidos para esta configuración, son los descritos en la sección 3.2.1 y los específicos de la memoria tabú, descritos a continuación:

- Tenure. Es el tiempo o número de iteraciones que un elemento (solución, movimiento o atributo) permanece en la lista tabú.

- Explícita global. Guarda las soluciones visitadas durante la ejecución del algoritmo, comportándose como una lista tipo FIFO (First In, First Out), cuando se supera el número máximo de soluciones almacenadas (en este, el tenure indica la longitud de la lista).

- Explícita por punto de captura. Almacena para cada punto de captura las soluciones visitadas durante su explotación. Funciona como una lista FIFO al exceder su capacidad de soluciones guardadas (en este caso, el tenure indica la longitud de la lista).

- Implícita por atributos. Para cada punto de captura penaliza los atributos (oraciones) que están presentes en el 50% de las soluciones visitadas recientemente, prohibiendo de forma temporal que se utilicen en la generación de una nueva solución candidata (en este caso el tenure, indica el número de intentos que estarán penalizados los atributos).

- Implícita por movimientos. Para cada punto de captura penaliza los movimientos entre atributos (intercambio de oraciones) que se realizan para generar una solución candidata, prohibiendo de forma temporal que se utilicen en la generación de una nueva solución (en este caso, el tenure indica el número de intentos que estarán penalizados los movimientos).

3.4.2 Afinación de parámetros del FSP

El proceso de afinación utiliza la configuración final de función objetivo (sección 3.1.6) y la configuración de los parámetros asociados al problema que se muestra en la Tabla 13. Este proceso evalúa al algoritmo FSP adaptando cada tipo de memoria tabú, sin optimizar y optimizando, teniendo en cuenta solamente el esquema C-Estático que presento mejores resultados que el C-Dinámico.

Sin Optimización. El Proceso de afinación sin optimización se realiza de forma similar a la afinación de parámetros del FSP (sección 3.1.7), diferenciándose al introducir el tipo de memoria tabú y utilizar únicamente el esquema C Estático. A continuación se presentan los resultados para cada tipo de memoria tabú:

- Explícita global. El valor del parámetro $tenure$ para este tipo de memoria se varía en incrementos de cinco, en el rango entre cero y ochenta. Al observar los mejores resultados obtenidos presentados en la Tabla 18, el valor de $tenure$ se establece en sesenta para el tipo Explícita Global.

Experimento	DUC2001			DUC2002		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
75	0,45983	0,20546	0,22824	0,48860	0,23051	0,24697
60	**0,45994**	**0,20561**	**0,22842**	**0,48867**	**0,23060**	**0,24704**
55	0,45973	0,20547	0,22820	0,48857	0,23049	0,24698
50	0,45980	0,20546	0,22824	0,48860	0,23056	0,24701

Tabla 18. Mejores Resultados Explícita Global.

- Explícita por punto de captura. El valor del parámetro $tenure$ y el rango es igual a la Explícita global. Los mejores resultados obtenidos son presentados en la Tabla 19, mostrando que $tenure$ también se estable en sesenta para el tipo Explícita por punto de captura.

Experimento	DUC2001			DUC2002		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
60	**0,46004**	**0,20582**	**0,22845**	**0,48862**	**0,23056**	**0,24705**
55	0,45943	0,20550	0,22812	0,48866	0,23054	0,24702
50	0,45924	0,20519	0,22787	0,48868	0,23060	0,24704
45	0,45973	0,20566	0,22823	0,48854	0,23051	0,24694

Tabla 19. Mejores Resultados Explícita Global por punto de captura.

- Implícita por atributos. El valor del parámetro $tenure$ para este tipo de memoria también varía de cinco en cinco, pero el rango se define entre cinco y treinta. Al observar los mejores resultados obtenidos presentados en la Tabla 20, el valor de $tenure$ se estable en cinco para el tipo Implícita por atributos.

Experimento	DUC2001			DUC2002		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
20	0,45925	0,20519	0,22784	0,48780	0,22981	0,24637
15	0,45918	0,20511	0,22782	0,48773	0,22972	0,24630
10	0,45953	0,20589	0,22838	0,48792	0,22993	0,24646
5	**0,45978**	**0,20564**	**0,22833**	**0,48800**	**0,23002**	**0,24652**

Tabla 20. Mejores Resultados Implícita por Atributos.

- Implícita por movimientos. El valor del parámetro $tenure$ y el rango es igual a la Implícita por atributos. Al observar los mejores resultados obtenidos presentados en la Tabla 21, $tenure$ también se estable en cinco para el tipo Implícita por movimientos.

Experimento	DUC2001			DUC2002		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
20	0,45959	0,20537	0,22800	0,48812	0,23001	0,24657
15	0,45954	0,20537	0,22802	0,48815	0,23003	0,24657
10	0,45966	0,20547	0,22810	0,48797	0,22983	0,24641
5	**0,45957**	**0,20559**	**0,22811**	**0,48806**	**0,23001**	**0,24654**

Tabla 21. Mejores Resultados Implícita por Movimientos.

Después de finalizado este proceso, se presentan los resultados obtenidos para cada tipo de memoria tabú en la Tabla 22, donde se observa que la memoria *Explícita por Punto de Captura* obtiene los mejores resultados en cuatro de medidas (ROUGE 1, ROUGE 2 y ROUGE SU4R sobre DUC2001 y ROUGE SU4R sobre DUC2002).

Experimento	DUC2001			DUC2002		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
Explícita Global	0,45994	0,20561	0,22842	**0,48867**	**0,23060**	0,24704
Explícita por Punto de Captura	**0,46004**	**0,20582**	**0,22845**	0,48862	0,23056	**0,24705**
Implícita por Atributos	0,45978	0,20564	0,22833	0,48800	0,23002	0,24652
Implícita por Movimientos	0,45957	0,20559	0,22811	0,48806	0,23001	0,24654

Tabla 22. Mejor Resultado de cada esquema de la memoria tabú.

Con Optimización: el proceso de afinación con optimización se realiza con cada algoritmo de búsqueda local (Ascenso de la Colina y Temple Simulado), de forma similar a la afinación de parámetros del FSP (secciones 3.2.2 y 3.3.2 respectivamente), pero agregando los parámetros del tipo de memoria tabú y utilizando únicamente el esquema C Estático. Luego de finalizar este proceso para cada uno de los algoritmos, se selecciona el mejor resultado que obtuvo cada uno de los esquemas de la memoria tabú luego de su evaluación, tal y como se presentan en la Tabla 23 para el algoritmo Ascenso de la Colina y en la Tabla 24 para el algoritmo Temple Simulado.

En la Tabla 23 se observa que el algoritmo Ascenso de la Colina con el esquema de optimización *OptMejorAlIterar* y una memoria *Explicita por Punto de Captura* obtiene los mejores resultados en cinco medidas (ROUGE 1 y ROUGE SU4 sobre DUC2001 y ROUGE 1, ROUGE 2 y ROUGE SU4 sobre DUC2002).

Experimento	DUC2001			DUC2002		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
Explícita Global - OptMejorAlIterar	0,45954	0,20549	0,22814	0,48816	0,23010	0,24665
Explícita por Punto de Captura - OptMejorAlIterar	0,45937	**0,20562**	**0,22825**	**0,48843**	**0,23033**	**0,24684**
Implícita por Movimientos - OptMejorAlIterar	**0,45991**	0,20594	0,22851	0,48791	0,22990	0,24644
Implícita por Atributos - OptMejorAlIterar	0,45967	0,20607	0,22852	0,48766	0,22978	0,24628

Tabla 23. Mejor Resultados FSP con Memoria tabú y Ascenso de la Colina

De otro lado, en la Tabla 24 se observa que el algoritmo Temple Simulado con el esquema de optimización *OptMejorAlIterar* y una memoria *Explicita por Punto de Captura*

obtiene los mejores resultados en todas las medidas ROUGE sobre DUC2001 y DUC2002.

Experimento	DUC2001			DUC2002		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
Explícita Global - OptMejorAlIterar	0,45934	0,20549	0,22797	0,48809	0,23009	0,24661
Explícita por Punto de Captura - OptMejorAlIterar	**0,45952**	**0,20575**	**0,22827**	**0,48843**	**0,23028**	**0,24680**
Implícita por Movimientos - OptMejorAlIterar	0,45944	0,20574	0,22828	0,48785	0,22988	0,24634
Implícita por Atributos - OptMejorAlIterar	0,45971	0,20588	0,22837	0,48798	0,23000	0,24650

Tabla 24. Mejor Resultado FSP con Memoria tabú y Temple Simulado

Al finalizar se presenta en la Tabla 25 el mejor resultado de cada una de las Tablas 22, 23 y 24. Donde se observa que la memoria *Explicita por Punto de Captura* sin aplicar optimización obtiene mejores resultados en todas las medidas ROUGE sobre DUC2001 y DUC2002 respecto a las otras combinaciones de tipos de memoria tabú y esquemas de optimización. Los valores de los parámetros del algoritmo FSP que corresponden a este experimento y que se constituyen como la mejor configuración encontrada en este ciclo, son los presentados en la Tabla 26.

Experimento	DUC2001			DUC2002		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
Explícita por Punto de Captura	**0,46004**	**0,20582**	**0,22845**	**0,48862**	**0,23056**	**0,24705**
Temple Simulado - Explícita por Punto de Captura - OptMejorAlIterar	0,45952	0,20575	0,22827	0,48843	0,23028	0,24680
Ascenso de la Colina - Explícita por Punto de Captura - OptMejorAlIterar	0,45937	0,20562	0,22825	0,48843	0,23033	0,24684

Tabla 25. Mejores resultados FSP con memoria tabú.

Parámetro	Valor
T	6
N	13
L	2
M	10
C Estático	1
Numero de Optimizaciones	0
Porcentaje de Puntos a Mejorar	0.0
Algoritmo de Optimización Local	Ninguno
Optimización	Ninguna
Memoria Tabú	Explícita por Punto de Captura

Tabla 26. Mejor Combinación de parámetros del FSP con una memoria tabú.

3.4.3 Esquema Algoritmo FSP-MT-SingleDocSum

El esquema del algoritmo FSP-MT-SingleDocSum presentado en la Figura 3.7 es la adaptación de una memoria tabú tipo explícita para cada punto de captura en el algoritmo FSP-SingleDocSum. El algoritmo propuesto en este ciclo difiere respecto al FSP-SingleDocSum (ver Figura 3.1) en que implementa listas tabú ($Inicializar(LT_i)$) que permiten almacenar para cada punto de captura (x_i) las soluciones que se crean a partir de su ajuste. Luego de crear y evaluar el punto de captura (x_i) se agrega a la lista tabú (LT_i). Además en el procedimiento $Generar\,Vecino()$ presentado en el esquema de la Figura 3.8 se agrega un ciclo condicional que verifica ($VericarMemoriaTabu(aux_i)$) si la solución reparada (aux_i) no se encuentra en la lista tabú (LT_i) del punto de captura (x_i), de manera que, si la solución (aux_i) está en la lista tabú ($Retorna\,Verdadero$) no se tiene en cuenta como punto de red por lo que se repiten las operaciones para obtener uno nuevo, si por el contrario, la solución no está en la lista tabú es tomada como un nuevo punto red y se agrega a la lista tabú (LT_i).

```
INICIO FSP
C = 1 /*Se estable el coeficiente de amplitud*/
Tenure = 60  /*Tamaño máximo de la tabú*/

HACER
    Inicializar x_i /*Crear el punto de captura i*/
    Inicializar LT_i /*Crear Lista Tabú para el punto de captura i*/
    Evaluar x_i /*Evaluar la calidad del punto de captura i*/
    Agregar x_i dentro de LT_i
    Instanciar p_i /*Establecer el punto de captura i como mejor solución local*/
    Actualizar G_best /*Actualizar la mejor solución global*/
HASTA Crear N Puntos de Captura

PARA t = 0 HASTA T HACER /*Numero de iteraciones*/
       PARA i = 0 HASTA N HACER /*Número de Puntos de Captura*/
              PARA l = 0 HASTA L HACER /*Numero de Lanzamientos*/
                  PARA k = 0 HASTA M HACER /*Número de Puntos de Red*/
                     y_k = Generar Vecino(x_i)    /*Generar Punto de Red*/
                     Evaluar y_k /*Evaluar la calidad del punto red k*/
                     Actualizar p_i /*Actualizar la mejor solución local*/
              FIN PARA
              Actualizar x_i /*Actualizar el punto de captura i*/
              Actualizar G_best /*Actualizar la mejor solución global*/
       FIN PARA
FIN PARA
RETORNAR G_best
```

Figura 3.7 Esquema Algoritmo FSP-MT-SingleDocSum

```
INICIO Generar Vecino

REPETIR
    aux_i = Copiar(x_i) /*Copiar el punto de captura i/*
    Puntuar Oraciones de (aux_i)
    Deshabilitar Oración en (aux_i)
    Habilitar Oración en (aux_i)
    Reparar Solución (aux_i)
HASTA VerificarMemoriaTabu(aux_i) = Falso /*Verifica si la solución es Tabú*/

RETORNAR aux_i
FIN Generar Vecinos
```

Figura 3.8 Esquema Procedimiento Generar Vecinos con Memoria Tabú

Finalizados los cuatros ciclos de diseño se presenta en la Tabla 27 el mejor resultado obtenido por el algoritmo propuesto en cada ciclo, donde se observa que el algoritmo con mejor desempeño fue FSP-MT-SingleDocSum debido a que mostro los mejores resultados en cuatro medidas (ROUGE SU4 sobre DUC2001 y ROUGE 1, ROUGE 2 y ROUGE SU4 sobre DUC2002) y en dos las segundas mejores (ROUGE 1 y ROUGE 2 sobre DUC2001), y por lo tanto se selecciona como propuesta definitiva.

Algoritmo	DUC2001			DUC2002		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
FSP-SingleDocSum	0,45982	0,20589	0,22852	0,48809	0,23015	0,24666
FSP-ASC-SingleDocSum	**0,46027**	**0,20627**	0,22784	0,48801	0,22992	0,24645
FSP-TS-SingleDocSum	0,4597	0,20604	0,22791	0,48799	0,22992	0,24639
FSP-MT-SingleDocSum	0,46004	0,20582	**0,22845**	**0,48862**	**0,23056**	**0,24705**

Tabla 27. Mejor Resultado de cada Ciclo.

Capítulo 4

4 ALGORITMO PROPUESTO: FSP-MT-SINGLEDOCSUM

En este capítulo se describe el nuevo algoritmo propuesto llamado FSP-MT-SingleDocSum, para la generación automática de resúmenes genéricos de un documento basado en el algoritmo FSP de Alejo-Machado [59] adaptando una memoria tabú explícita. En las siguientes secciones se presenta: la representación de las soluciones escogida, la función objetivo utilizada para evaluar la calidad de los resúmenes generados (solución candidata), la descripción de los componentes del algoritmo FSP-MT-SingleDocSum y el esquema general del proceso para generar un resumen.

4.1 REPRESENTACIÓN DE LAS SOLUCIONES

Los algoritmos metaheurísticos trabajan directamente sobre una abstracción de los objetos solución, debido a esto, es importante definir su representación dentro del espacio de búsqueda del problema. La representación se refiere al mecanismo para definir la estructura a utilizar en la generación, ajuste y evaluación de una solución candidata [74]. Esta estructura debe permitir mapear cada punto del espacio de búsqueda en una solución candidata (codificación) y obtener desde la solución candidata el punto del espacio que está representando (descodificación). De este modo la codificación se utiliza en los procedimientos de construcción y ajuste de las soluciones, mientras que la decodificación se usa en la evaluación de las soluciones candidatas y en la transformación de la solución final.

Considerando los aspectos anteriores, en FSP-MT-SingleDocSum, la representación de una solución es realizada mediante codificación binaria que se caracteriza por su simplicidad y flexibilidad para codificar y trabajar con problemas de múltiples variables, la fácil manipulación por parte de los distintos operadores que intervienen en el proceso de búsqueda y la facilidad para ser decodificada en la solución real del problema [51]. La codificación binaria utiliza un vector binario, en este caso, el tamaño del vector es igual al número de oraciones (*n*) que componen el documento representado como, $\{s_1, s_2, \dots, s_n\}$ y cada elemento del vector toma el valor de uno o cero para representar la presencia o ausencia de una oración del documento en el resumen. Por ejemplo, si un documento tiene ocho oraciones $(n = 8)$, la representación de un vector solución puede ser la siguiente $[1,0,1,0,1,0,1,0]$, en este caso, indicando que las oraciones primera, tercera, quinta y séptima están habilitadas para ser parte de la solución candidata (resumen candidato) [51, 79].

La decodificación consiste en buscar ordenadamente las oraciones cuya posición en el documento coincide con la posición de los elementos del vector solución que tengan valor uno, para así obtener las oraciones que conforman el resumen candidato. De este modo, la evaluación de una solución candidata y la conformación del resumen final se realiza con las oraciones obtenidas después de la decodificación [51].

4.2 FUNCIÓN OBJETIVO

La función objetivo a optimizar por el algoritmo FSP-MT-SingleDocSum está compuesta por características que dependen de la oración como: posición en el documento, similitud con el título y longitud (medida en palabras); y otras características que dependen de la similitud con las oraciones del resumen candidato como: cohesión que es la similitud entre las oraciones del resumen y cobertura que es la similitud de las oraciones del resumen con respecto al documento. Estas características fueron calculadas con las ecuaciones: *Posición 2* (3.2), *Relación con el Título* (3.6), *Longitud 2* (3.7), *Cohesion 2* (3.13) y *Cobertura 2* (3.15), tal y como se presenta en la Sección 3.1.6. De este modo la función objetivo es calculada de acuerdo a la Ecuación (4.1), con su correspondiente fórmula matemática en la Ecuación (4.2).

$$f(x) = 0.15 * Posición + 0.05 * Relación\ con\ el\ título + 0.1 * Longitud + 0.05 * Cohesión + 0.65 * Cobertura \tag{4.1}$$

$$f(x) = \sum_{\forall s_i \in Summary} \left[0.15 * \frac{PositionsRanking(s_i)}{O} + 0.05 * \frac{sim(s_i, t)}{máximo_{\forall\ summary} TR} + 0.1 * \frac{\frac{1 - e^{-\frac{l(s_i) - \mu(l)}{std(l)}}}{1 + e^{-\frac{l(s_i) - \mu(l)}{std(l)}}}}{N} + 0.05 * \sum_{\forall s_j \in Summary} \frac{sim_{cos}(s_i, s_j)}{N_s} \right] + 0.65 * sim_{cos}(R, D) \tag{4.2}$$

Como se observa en la Ecuación (4.1) cada uno de los factores presentes en esta función objetivo va acompañado de un coeficiente utilizado para el proceso de afinamiento de la función objetivo. Con la restricción que la suma de estos coeficientes debía ser igual en cualquiera de las combinaciones establecidas.

4.3 ADAPTACIÓN FSP CON MEMORIA TABÚ

Las modificaciones que se realizaron al algoritmo FSP original para adaptarlo al problema de la generación automática de resumes de texto fueron las siguientes:

- Se estableció la representación binaria de las soluciones candidatas (en FSP es continua).
- El parámetro coeficiente de amplitud (C) fue adaptado para permitir el ajuste de las soluciones con representación binaria, de forma que indique el número de oraciones a modificar en el punto de captura (ver sección 3.1.3). Además al finalizar los ciclos de diseño se estableció en un valor estático (uno), por lo tanto no es necesario aplicar la estrategia de actualización para este parámetro.

- El procedimiento para crear un punto de red (y_k) se lleva a cabo en $Generar\ Vecino()$.
- La adaptación de una memoria tabú tipo explícita para cada punto de captura (x_i), a través de la implementación de listas tabú $(Inicializar(LT_i))$ que permiten almacenar para cada punto de captura (x_i) las soluciones que se generan a partir de su ajuste.
- Al retornar la mejor solución global G_{best} el algoritmo FSP-MT-SingleDocSum ordena las oraciones según el *Criterio de Selección de Oraciones al generar el Resumen* para que aparezcan en ese orden en el resumen generado.

El esquema del algoritmo FSP-MT-SingleDocSum es presentado en la Figura 4.1.

```
INICIO FSP
C = 1 /*Se estable el coeficiente de amplitud*/
Tenure = 60  /*Tamaño máximo de la tabú*/

HACER
    Inicializar x_i /*Crear el punto de captura i*/
    Inicializar (LT_i) /*Crear Lista Tabú para el punto de captura i*/
    Evaluar x_i /*Evaluar la calidad del punto de captura i*/
    Agregar x_i dentro de (LT_i)
    Instanciar p_i /*Establecer el punto de captura i como mejor solución local*/
    Actualizar G_best /*Actualizar la mejor solución global*/
HASTA Crear N Puntos de Captura

PARA t = 0 HASTA T HACER /*Numero de iteraciones*/
       PARA i = 0 HASTA N HACER /*Número de Puntos de Captura*/
              PARA l = 0 HASTA L HACER /*Numero de Lanzamientos*/
                  PARA k = 0 HASTA M HACER /*Número de Puntos de Red*/
                     y_k = Generar Vecino(x_i)    /*Generar Punto de Red*/
                     Evaluar y_k /*Evaluar la calidad del punto red k*/
                     Actualizar p_i /*Actualizar la mejor solución local*/
              FIN PARA
              Actualizar x_i /*Actualizar el punto de captura i*/
              Actualizar G_best /*Actualizar la mejor solución global*/
       FIN PARA
FIN PARA
RETORNAR G_best
```

Figura 4.1 Esquema Algoritmo FSP-MT-SingleDocSum

El procedimiento $Generar\ Vecino()$ presentado en la Figura 4.2 a diferencia del algoritmo FSP efectúa algunas tareas adicionales para crear un punto de red:

- Inicialmente se genera una solución copia (aux_i) del punto de captura a ajustar (C$opiar(x_i)$).
- Se evalúan las oraciones en la solución copia (aux_i) de acuerdo a las características establecidas en el *Criterio para Deshabilitar una oración* $(PuntuarOraciones(x_i))$.
- Se selecciona la oración con la menor evaluación para ser deshabilitada $(Deshabilitar\ Oración\ en\ (aux_i))$.

- A continuación se habilita aleatoriamente una oración en la solución copia (aux_i) $(Habilitar\ Oración\ en\ (aux_i))$.
- Después se realiza una comprobación y ajuste para que la solución copia (aux_i) cumpla con el límite de palabras del resumen establecido por el parámetro asociado al problema *Máxima longitud a Evaluar de un Resumen* $(Reparar\ Solucion\ (aux_i))$.
- Verifica $(VericarMemoriaTabu(aux_i))$ si la solución luego de ser reparada (aux_i) no se encuentra en la lista tabú (LT_i) del punto de captura (x_i), de manera que, si la solución (aux_i) está en la lista tabú $(Retorna\ Verdadero)$ no se tiene en cuenta como punto de red y por lo tanto se deben repetir las tareas ya mencionadas para crear uno nuevo, si por el contrario, la solución no está en la lista tabú es tomada como un nuevo punto red y es agregada a la lista tabú (LT_i).
- Al final retorna la solución copia (aux_i), en otras palabras el nuevo punto de red y_k

```
INICIO Generar Vecino

REPETIR
    aux_i = Copiar(x_i) /*Copiar el punto de captura i/*
    Puntuar Oraciones (aux_i)
    Deshabilitar Oración (aux_i)
    Habilitar Oración (aux_i)
    Reparar Solución (aux_i)
HASTA VerificarMemoriaTabu(aux_i) = Falso /*Verifica si la solución es Tabú*/

RETORNAR aux_i
FIN Generar Vecinos
```

Figura 4.2 Procedimiento Generar Vecinos FSP

4.3.1 Condición de parada

La ejecución del algoritmo termina cuando la ejecución de FSP-MT-SingleDocSum realiza las *T* iteraciones y los *L* lanzamientos sobre cada uno de los *N* puntos de captura.

4.3.2 Generación de las Soluciones Iniciales

Inicialmente (tiempo $t = 0$) se generan N puntos de captura (x_i) que se representan de la siguiente forma, de acuerdo a la Ecuación (4.3).

$$x_i(0) = \left[x_{i,1}(0), x_{i,2}(0), \dots, x_{i,s}(0), \dots, x_{i,n}(0)\right] \quad (4.3)$$

Donde n es el número de oraciones en el documento, $x_{i,s}(0) \in \{0,1\}$ es un entero binario, $i = 1,2, \dots, N$, N es el número de puntos de captura.

El proceso para crear cada x_i, en primer lugar selecciona aleatoriamente una oración del documento, para que todas las oraciones del documento tengan la misma probabilidad de pertenecer al resumen candidato, luego verifica que la longitud de la oración en palabras no sobrepase el límite de número de palabras del resumen, si cumple esta restricción, se

habilita esta oración en x_i. Este proceso se repite hasta que se cumpla el límite de número de palabras del resumen de acuerdo a la Ecuación (4.4).

$$x_{i,a}(0) = \begin{cases} 1, Sí & \sum_{s_j \in x_i} l_j + l_a \leq L_{MAX} \\ 0, & De\ otro\ modo \end{cases} \quad (4.4)$$

Donde $i = 1,2,\dots,N$, N es el número de puntos de captura, a es un numero entero aleatorio entre $[1,n]$, n es el número de oraciones en el documento, $x_{i,a}(0) \in \{0,1\}$ es un entero binario, l_j es la longitud (medida en palabras) de la *j*-ésima oración del resumen, l_a es la longitud (medida en palabras) de la oración escogida según a y L_{MAX} es el límite máximo de numero de palabras.

Después de generar cada $x_i(0)$, se establece inicialmente como la mejor solución local, de acuerdo a la Ecuación (4.5) y se verifica si $x_i(0)$, tiene mejor calidad que la mejor solución global (G_{best}) (ver sección 4.3.7).

$$p_i = x_i(0), \forall x_i(0) \in X \quad (4.5)$$

Donde $i = 1,2,\dots,N$, N es el número de puntos de captura, p_i es la mejor solución local de $x_i(0)$ y X es el conjunto de puntos de captura.

4.3.3 Ajuste de C

El valor de C, es importante porque permite graduar el ajuste sobre x_i, es decir, permite controlar que tan diferentes serán las soluciones generadas a partir de x_i. En el algoritmo propuesto el valor de C determina cuantos elementos se pueden cambiar (número de oraciones que se adicionan y eliminan) de x_i, de modo que, si el valor de C es alto con respecto al tamaño de la solución (número de oraciones), el ajuste es fuerte y por lo tanto las soluciones generadas a través de este ajuste, serán muy diferentes respecto a la solución original. Si su valor es bajo, el ajuste será mínimo y las soluciones generadas a través de este ajuste, serán similares a la solución original. En este caso el valor de C es estático y pertenece al conjunto de los números enteros.

4.3.4 Generar Vecino

Con cada punto de captura x_i, se realiza un numero de lanzamientos de la red de pesca (L), que representa el número de vecindarios que se van a generar por cada x_i. En cada lanzamiento (l) se generan puntos de red (M), que representan el número de vecinos que se generan a través del ajuste de x_i.

Para generar cada punto de red y_k, inicialmente se realiza una copia de x_i en una variable temporal (aux_i) y luego se realiza un proceso de ajuste que comprende las siguientes tareas: puntuar oraciones, deshabilitar oración, habilitar oración, reparar solución y Verificar Solución Tabú. El número de elementos de x_i que serán ajustados, es definido por el coeficiente de amplitud (C), de esta forma las tareas deshabilitar oración y habilitar oración se repiten C veces.

Las tareas llevadas a cabo en este procedimiento, se describen en las siguientes secciones.

4.3.4.1 Puntuar Oraciones

Esta tarea consiste en evaluar cada una de las oraciones habilitadas en aux_i y obtener el índice de la oración con el puntaje más bajo (d), de acuerdo a la Ecuación (4.6).

$$d = \begin{Bmatrix} s, Sí\ aux_{i,s} = 1\ \text{^}\ aux_{i,d} = 1\text{^}\ PuntuarOracion(s) < PuntuarOracion(d) \\ d, \qquad\qquad De\ otro\ modo \end{Bmatrix} \quad (4.6)$$

Donde $d \in [1, n]$ (n es el número de oraciones del documento) representa el índice de la oración a con menos puntaje de aux_i, $aux_{i,s}$ representa la *s*-ésima oración en aux_i y $s = 1,2, \dots, n$.

El puntaje $PuntuarOracion(s)$ de la *s*-ésima oración se obtiene al sumar el ranking de posición de la oración con el valor obtenido al evaluar la similitud de cosenos entre la oración y el documento (cobertura), como se presenta en la Ecuación (4.7).

$$PuntuarOracion(s) = \frac{2 - 2 * \left(\frac{pos_s - 1}{n - 1} \right)}{n} + sim_{\cos}(V_s, D) \quad (4.7)$$

Donde $s = 1,2, \dots, n,\ u = 1,2, \dots, n,$ (n es el número de oraciones del documento) s y u representa el índice de la *s*-ésima y *u*-ésima oración del documento, respectivamente, pos_s representa la posición de la oración s en el documento, V_s y D son los vectores de términos que representan la oración s y al documento, respectivamente.

4.3.4.2 Deshabilitar Oración

Esta tarea consiste en deshabilitar en aux_i la oración (d) con el puntaje más bajo de acuerdo a la Ecuación (4.8).

$$aux_{i,s} = \begin{Bmatrix} 0, & Sí\ \ s = d \\ aux_{i,s}, & De\ otro\ modo \end{Bmatrix} \quad (4.8)$$

Donde $aux_{i,s}$ representa la *s*-ésima oración del *i*-ésimo punto de captura, $i = 1,2, \dots, N$, (N es el número de puntos de captura), $s = 1,2, \dots, n$, (n es el número de oraciones del documento), $d \in [1, n]$, representa el índice de la oración a deshabilitar en aux_i.

4.3.4.3 Habilitar Oración

Esta tarea consiste en seleccionar aleatoriamente una oración q del documento y habilitarla en aux_i, de acuerdo a la Ecuación (4.9).

$$aux_{i,s} = \begin{Bmatrix} 1, & Sí\ \ s = d \\ aux_{i,s}, & De\ otro\ modo \end{Bmatrix} \quad (4.9)$$

Donde $aux_{i,s}$ representa la *s*-ésima oración del *i*-ésimo punto de captura, $i = 1,2,\dots,N$, (N es el número de puntos de captura), $s = 1,2,\dots,n$, (n es el número de oraciones del documento), $q \in [1,n]$, representa el índice de la oración a habilitar en aux_i.

4.3.4.4 Reparación de la Solución

Este procedimiento muy similar al proceso de generación de una solución inicial, selecciona aleatoriamente una oración habilitada en aux_i y verifica que su longitud en palabras sumada a la longitud en palabras del resumen, no sobrepase el límite de número de palabras del resumen, si cumple esta restricción la habilita en y_k y si no cumple la restricción no la habilita. Este proceso se repite hasta que se cumpla el límite de número de palabras del resumen, como se muestra en la Ecuación (4.10).

$$y_{i,a} = \begin{Bmatrix} 1, Sí \;\; aux_{i,a} = 1 \,\hat{}\, \sum_{s_j \in aux_i} l_j + l_a \leq L_{MAX} \\ 0, \qquad\qquad De\ otro\ modo \end{Bmatrix} \quad (4.10)$$

Donde a es un numero entero aleatorio entre $[1,n]$, n es el número de oraciones en el documento, $y_{k,a} \in \{0,1\}$ es un entero binario, $i = 1,2,\dots,N$, N es el número de puntos de captura, l_j es la longitud (medida en palabras) de la *j*-ésima oración del resumen, es la longitud (medida en palabras) de la oración escogida según a y L_{MAX} es el límite máximo de numero de palabras.

4.3.4.5 Verificar Memoria Tabú

El algoritmo FSP-MT-SingleDocSum adapta una memoria tabú de tipo explícita a través de la implementación de una lista (LT_i) para cada punto de captura (x_j) que almacena los vecinos generados (puntos de red) a partir de su ajuste, con el objetivo de incentivar la diversificación de las soluciones candidatas en el proceso de generación de los puntos de red. Esta memoria almacena sesenta soluciones candidatas $(tenure = 60)$, una vez se supera este límite, la memoria actúa como una lista de tipo FIFO (Primero en Entrar, Primero en Salir), como se presenta en la Ecuación (4.11).

$$LT_i = [y_1, y_2, \dots, y_{tabu}, \dots, y_{tenure}], tabu = 1,2,\dots,tenure \quad (4.11)$$

Donde LT_i representa la lista tabú explícita para el punto de captura x_i, y_{tabu} representa un punto de red que está en la lista tabú y $tenure$ indica el número máximo de soluciones que puede almacenar la lista tabú.

Esta tarea verifica si el punto de red (y_k) es o no una solución tabú, de acuerdo a la Ecuación (4.12). Si el resultado de la Ecuación (4.12) es verdadero, la solución es tabú, y por lo tanto este punto de red no se toma en cuenta ya que ha sido visitado con anterioridad. De modo contrario el punto de red no es solución tabú, siendo válido como nueva solución candidata (nuevo vecino) y se agrega a la lista tabú (LT_i).

$$VeficarMemoriaTabu(y_k) = \begin{Bmatrix} Verdadero, & Sí \;\; y_k \in LT_i \\ False, & De\ otro\ modo \end{Bmatrix} \quad (4.12)$$

Donde y_k representa el punto de red a verificar, (LT_i) representa la lista tabú explícita para el punto de captura x_i presentada en la Ecuación (4.11).

A continuación se realiza la tarea Actualización de la Mejor Solución Local (ver sección), si la solución no es tabú. De lo contrario el proceso de ajuste se realiza desde el principio.

4.3.5 Actualización de la Mejor Solución Local

Una vez generado un nuevo punto de red (y_k), se verifica si la evaluación de y_k es mayor que la evaluación de la mejor solución local (p_i), si es así, p_i se actualiza con y_k, de lo contrario p_i se mantiene. Esta tarea se muestra en la Ecuación (4.13).

$$p_i = \begin{Bmatrix} y_k, Sí \quad f(y_k) > f(p_i) \\ p_i, \quad De\ otro\ modo \end{Bmatrix} \quad (4.13)$$

Donde y_k representa la *k*-ésima solución candidata generada a través del ajuste de x_i, $i = 1,2,\dots,N$, N es el número de puntos de captura, k= $1,2,\dots,M$, M es el número de puntos de red, $l = 1,2,\dots,L$, L es el número de lanzamientos, $f(p_i)$ y $f(y_k)$ representa la evaluación de p_i y y_k con la Ecuación (4.2), respectivamente.

A continuación, si no se han generan M puntos de Red o no se han efectuado los (L) lanzamientos se realiza la tarea Generar Vecino (ver sección 4.3.4). De lo contrario se lleva a cabo la tarea Actualización del Punto de Captura (ver sección 4.3.6).

4.3.6 Actualización del Punto de Captura

Para cada punto de captura, se generan L lanzamientos y luego este proceso se itera T veces. Al final de cada iteración t, se verifica si la evaluación de la mejor solución local (p_i) es mayor a la evaluación del punto de captura (x_i), si esto cumple el x_i se actualiza con p_i, de acuerdo a la Ecuación (4.14).

$$x_i = \begin{Bmatrix} p_i, Sí \quad f(p_i) > f(x_i) \\ x_i, \quad De\ otro\ modo \end{Bmatrix} \quad (4.14)$$

Donde x_i representa el *i*-ésimo punto de captura, $i = 1,2,\dots,N$, N es el número de puntos de captura, p_i es la mejor solución encontrada en el vecindario de x_i, $f(p_i)$ y $f(x_i)$ representan la evaluación obtenida por p_i y x_i, en la Ecuación (4.2) respectivamente.

4.3.7 Actualización de la Mejor Solución Global

Cuando se encuentra que x_i tiene mejor evaluación que G_{best} se reemplaza la información actual de G_{best} por la contenida en x_i, de otro modo G_{best} se mantiene, como se muestra en la Ecuación (4.15).

$$G_{best} = \begin{Bmatrix} x_i(t), Sí \quad f(x_i(t)) > f(G_{best}) \\ G_{best}, \quad De\ otro\ modo \end{Bmatrix} \quad (4.15)$$

Donde $x_i(t)$ representa la *i*-ésimo punto de captura, $i = 1,2,\dots,N$, N es el número de puntos de captura, $t = 1,2,\dots,T$, T es el número de iteraciones, G_{best} es la memoria de la

mejor solución global, $f(G_{best})$ y $f(x_i(t))$ representan el puntaje obtenido al evaluar G_{best}best y $x_i(t)$ en la Ecuación (4.2) respectivamente.

4.4 ESQUEMA DE GENERACIÓN DE RESÚMENES

El proceso de generación de un resumen presentado en la Figura 4.3 inicia con el pre-procesamiento del documento original, en el cual, se segmenta el texto para obtener las oraciones que lo conforman, se normalizan dichas oraciones eliminando mayúsculas y palabras vacías, se llevan las palabras con la misma raíz a una forma común, y finalmente las oraciones son indexadas en una estructura de datos (Las tareas de pre-procesamiento son detalladas en la sección 6.1).

El siguiente paso del proceso consiste en tomar los términos de cada una de las oraciones obtenidas y llevarlas a la representación del modelo espacio vectorial descrito en la Sección 2.2.1, de esta manera para cada término se calcula su peso en función de su frecuencia relativa para que sea almacenado en una matriz de pesos, así mismo se ponderan los términos correspondientes al título del documento.

Seguidamente, con base en los pesos calculados, se determina la similitud de cosenos entre oraciones, la similitud de cosenos de cada oración y el documento, y la similitud de cosenos de cada oración con el título para que sean almacenados en una matriz de similitudes. Dichos valores de similitud servirán posteriormente para el cálculo de aquellos factores de la función objetivo que los requieran, como la cohesión, cobertura y relación con el título.

El último paso corresponde a la ejecución del algoritmo FSP-MT-SingleDocSum como fue descrito en la sección anterior, obteniendo al final un vector solución, cuyas posiciones con valor igual a uno son ordenadas descendentemente según su posición. Posteriormente el vector solución es decodificado para obtener las oraciones originales del documento respectivas, que finalmente conforman el resumen generado, el cual es truncado a cien palabras para realizar la evaluación de calidad de los resúmenes con otros métodos del estado del arte.

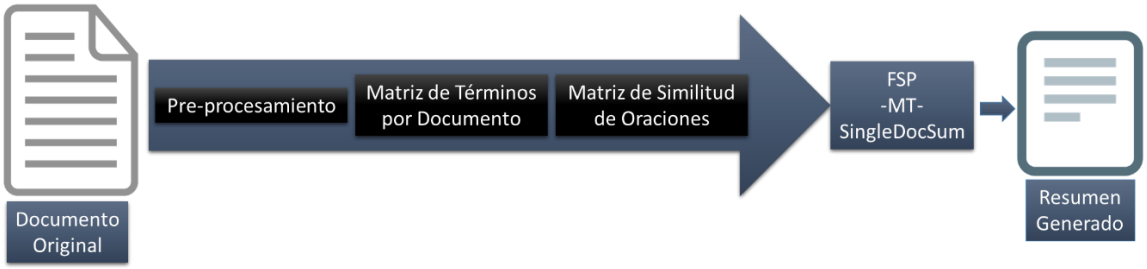

Figura 4.3 Esquema de Generación de Resúmenes

Capítulo 5

5 COMPLEMENTO DE WORD: FSP-MT-Summarizer

En este capítulo se presenta el complemento FSP-MT-SUMMARIZER para el programa ofimático Office Word, que permite generar resúmenes de documentos utilizando el algoritmo FSP-MT-SingleDocSum que fue diseñado y evaluado para generar automáticamente resúmenes genéricos de un documento en idioma inglés. En las siguientes secciones se presentan: la arquitectura, diagrama de clase e interfaz del complemento.

5.1 ARQUITECTURA DE FSP-MT-SUMMARIZER

Para la construcción de FSP-MT-Summarizar se utiliza Visual Studio Tools Office (VSTO) que permite personalizar las aplicaciones de Office con las características específicas que se necesiten. En el desarrollo y compilación de un complemento, se crea un ensamblado de código administrado que es cargado por una aplicación de Microsoft Office para responder a los eventos que se producen en la aplicación como, por ejemplo, cuando un usuario hace clic en un elemento de menú. El código del complemento de VSTO también puede hacer llamadas en el modelo de objetos para automatizar y extender la aplicación, además puede usar cualquiera de las clases de .NET Framework. La Figura 5.1 presenta la arquitectura de los complementos VSTO.

Figura 5.1 Arquitectura de FSP-MT-Summarizer

Se definió la arquitectura en N-Capas para construir el ensamblado del FSP-MT-Summarizer que es el componente principal de esta solución, ya que posibilita la distribución de componentes (capas), para obtener una separación de responsabilidades

de la interfaz de usuario y la lógica del negocio. En este proyecto se utilizan dos capas, de la siguiente manera:

- Capa de presentación: es la responsable de la interacción con el usuario y por lo tanto donde se implementan los mecanismos para obtener el texto del documento que se va a resumir y los datos de la configuración de parámetros del algoritmo FSP-MT-SingleDocSum.

- Capa lógica de negocio: se encarga de utilizar los datos obtenidos por el requerimiento del usuario para llevar a cabo el proceso de generación automática de resúmenes mediante el algoritmo FSP-MT-SingleDocSum.

5.2 DIAGRAMA DE CLASES

Se presenta en Figura 5.2 el diagrama de clases que representa a las entidades de las capas de presentación y lógica de negocio del FSP-MT-Summarizer, con sus atributos y métodos utilizados. A continuación se realiza una breve descripción de cada una de las clases:

- Interfaz: es una especialización de RibbonClase, una clase provista por VSTO para generar una interfaz de usuario tipo cinta de opciones (ribbon), que facilita la creación e integración de la interfaz del complemento al programa Office Word. Sus tareas consisten en obtener el texto del documento, crear y llamar el formulario de opciones para obtener los datos de los parámetros establecidos por el usuario y por último enviar esta información a la clase ControladorFSP-MT-SingleDocSum.

- FrmOpciones: clase de la capa de presentación, permite definir y crear un formulario para que el usuario establezca los parámetros FSP, de la función objetivo, asociados al problema, de memoria tabú y optimización local del algoritmo FSP-MT-SingleDocSum.

- ControladorFSP-MT-SingleDocSum: esta clase estática define los métodos que dan acceso al ensamblado FSP-MT-SingleDocSum.dll que contiene toda la lógica de la generación automática de resúmenes extractivos, incluyendo: el algoritmo FSP-MT-SingleDocSum, lógica de pre-procesamiento de textos y modelos vectoriales para la construcción de la matrices de términos por documento y similitud de oraciones. Los métodos de esta clase permiten establecer la configuración de parámetros del algoritmo FSP-MT-SingleDocSum con los datos provenientes de la Interfaz, para generar el resumen del texto del documento que está activo.

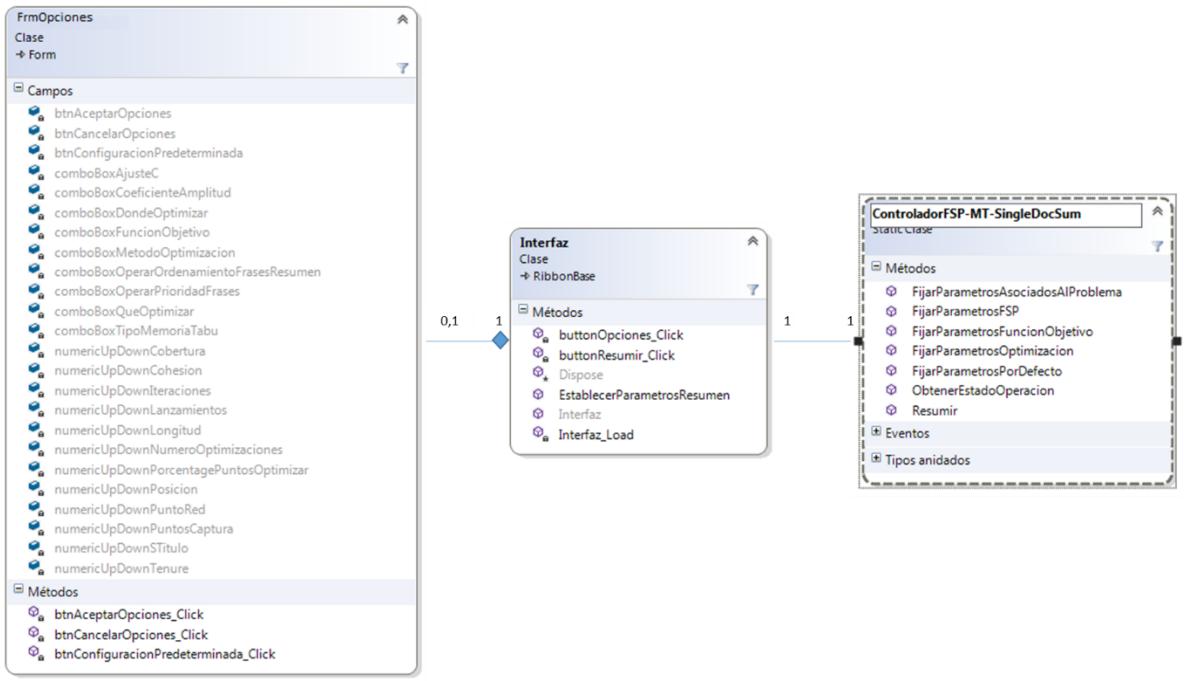

Figura 5.2 Diagrama de Clases de FSP-MT-Summarizer

5.3 INTERFAZ GRAFICA

Se presenta la interfaz gráfica utilizada por FSP-MT-Summarizer, que está compuesta por una cinta de opciones (ribbon) integrada al programa Office Word y un formulario de opciones para configurar los parámetros del algoritmo FSP-MT-SingleDocSum. A continuación se describen los componentes de la interfaz:

- Cinta de opciones FSP-MT-Summarizer: en la Figura 5.3 se observa la pestaña que da acceso al complemento (área demarcada en verde) y la cinta de opciones (área demarca en azul) con los botones *Resumir* y *Opciones*, además del campo de texto *Resumir documento al*.

- Formulario de Opciones: al dar clic en el botón *Opciones* de la anterior interfaz se abre un formulario que permite al usuario modificar la configuración actual del algoritmo FSP-MT-SingleDocSum. Como se observa en la Figura 5.4, se presentan cuatro pestañas que agrupan y permite modificar los parámetros: del algoritmo FSP, de la función objetivo, asociados al problema y de optimización local (incluye memoria tabú). Este formulario cuenta con controles específicos para la captura de datos numéricos y menús de opciones que evitan al usuario ingresar información errónea. También incluye los botones de: *Restablecer Configuración Predeterminada* que permite restaurar la configuración de los parámetros definitiva del algoritmo propuesto; *Aceptar* que acepta los cambios, guarda los datos de todos los parámetros y cierra el formulario; *Cancelar* que cierra el formulario sin guardar ningún cambio.

- *Resumir documento al:* este campo de texto permite definir el tamaño del resumen (dado en porcentaje) con respecto al documento original.

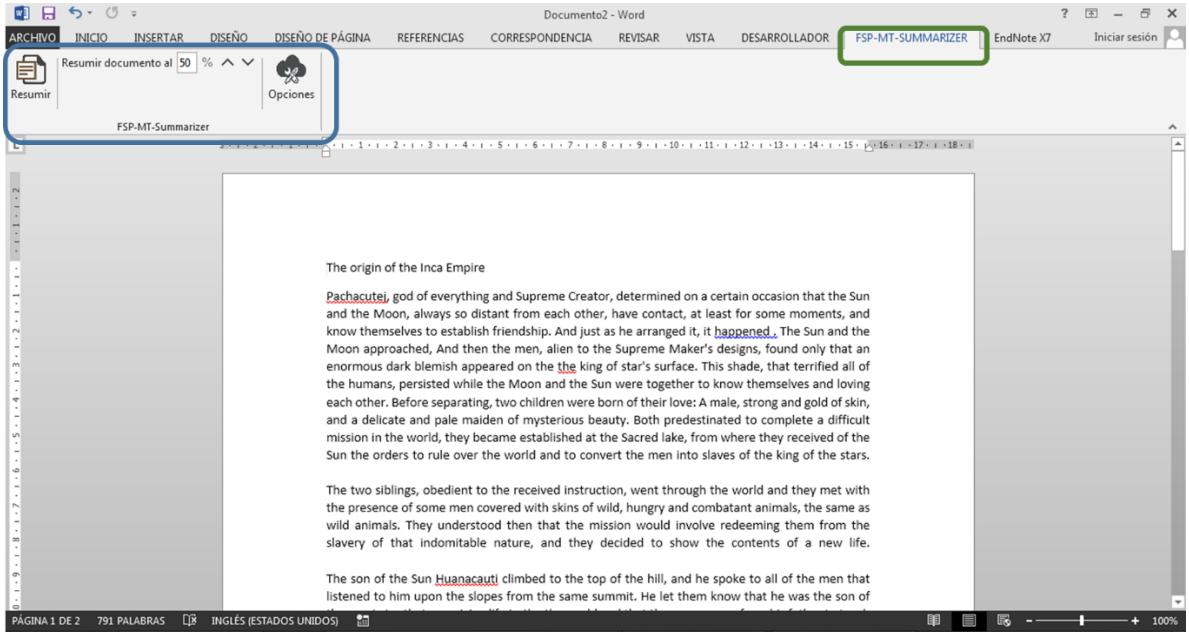

Figura 5.3 Cinta de Opciones FSP-MT-Summarizer

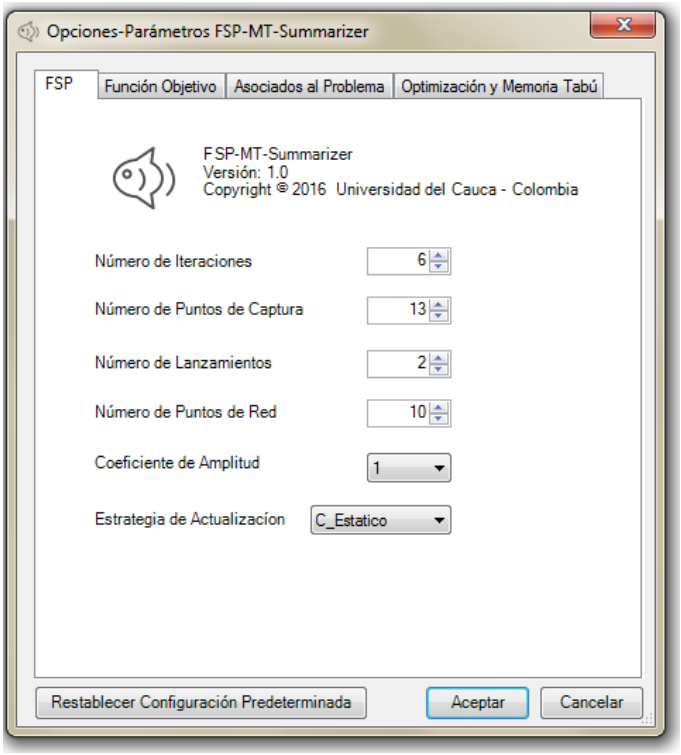

Figura 5.4 Formulario de Opciones FSP-MT-Summarizer

- Resumir: al dar clic en este botón, todo el texto del documento y los parámetros guardados son enviados para generar el resumen por medio del algoritmo FSP-MT-SingleDocSum. Una vez se resuma el documento se inserta el texto del resumen resaltado al final del documento, como se observa en la Figura 5.5.

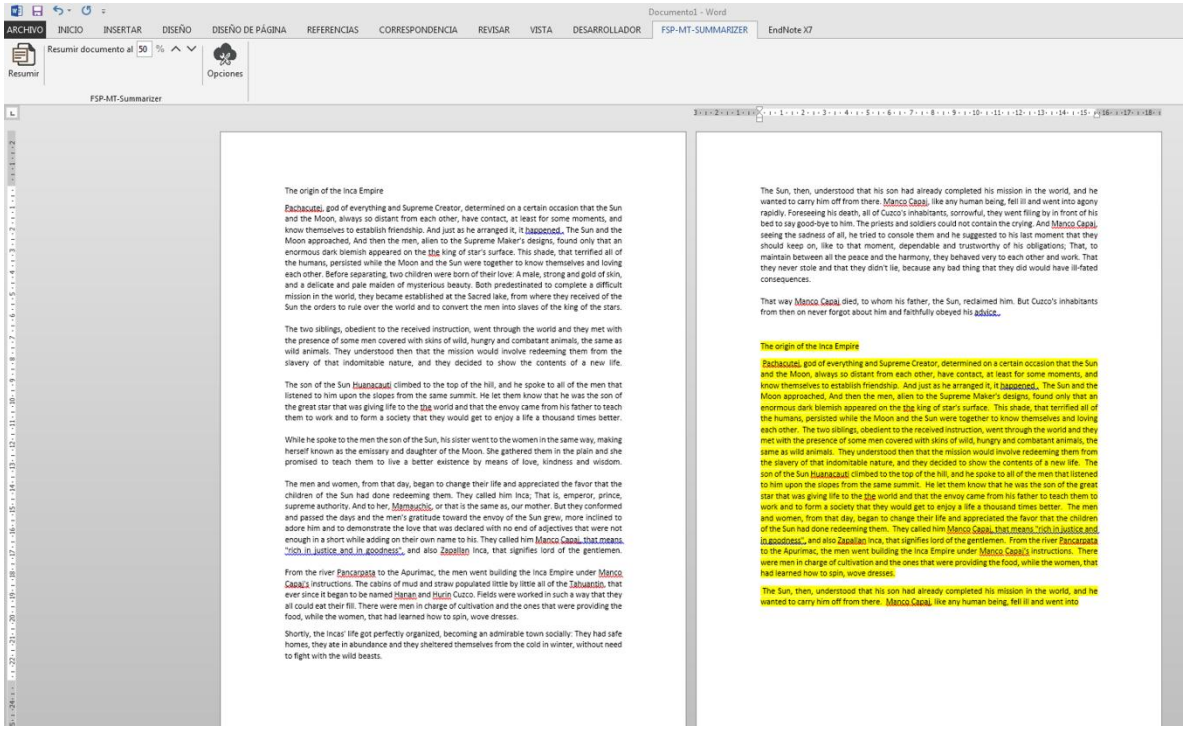

Figura 5.5 Presentación del resumen FSP-MT-Summarizer

Capítulo 6

6 EVALUACIÓN

En este capítulo se describen las tareas de pre-procesamiento del documento, la descripción de la colección de los documentos y de las métricas usadas en la experimentación. También el afinamiento de parámetros del algoritmo propuesto y por último los resultados obtenidos con los resúmenes generados por los esquemas planteados en los cuatro ciclos de desarrollo del algoritmo propuesto FSP-MT-SingleDocSum, y la comparación con los métodos del estado del arte.

6.1 NORMALIZACIÓN E INDEXACIÓN DE DOCUMENTOS

La *normalización* del texto tiene como propósito evitar la pérdida de información relevante en los datos presentados de manera informal, en concreto, para eliminar los saltos de línea adicionales, segmentar el texto en párrafos, eliminar espacios adicionales y los signos de puntuación extras, borrar señales innecesarias, restauran las palabras mal escritas, identificar los límites de la oración, entre otras [63].

En la generación automática de resúmenes de texto la normalización se utiliza para reducir los términos a una forma canónica[6] que permita la agrupación de términos conceptualmente relacionados, también porque permite la indexación, identificando los términos clave que serán utilizados por el sistema, lo que facilita la búsqueda y el ordenamiento [68]. En las siguientes secciones, se describen cada uno de los procesos de normalización e indexación ejecutados dentro del sistema propuesto en la presente investigación.

6.1.1 Segmentación

El proceso de segmentación consiste en dividir el texto en unidades significativas más manejables como palabras u oraciones, con fin de facilitar la recuperación y evaluación los elementos de un texto para determinar su valor e importancia. En la presente investigación se hace uso de una herramienta de segmentación de fuente abierta denominada "*splitta*"[7], la cual utiliza un enfoque estadístico que intenta lidiar con los problemas de ambigüedad en la detección de los límites de las oraciones de un texto, y cuyo desempeño ha sido reportado en un estándar de *Wall Street Journal*, con buenos resultados [88].

[6] La forma canónica de una palabra hace referencia a su forma estándar que, por convención, representa a todas sus flexiones. Esta forma canónica varía según el idioma, por ejemplo, los verbos en inglés se representan mediante la raíz no flexionada, mientras en francés o español se representan con el infinitivo del verbo.
[7] Esta herramienta se encuentra disponible en *http://code.google.com/p/splitta*.

6.1.2 Eliminación de mayúsculas y signos ortográficos

La conversión de mayúsculas a minúsculas y la eliminación de signos ortográficos, como tildes o diéresis, normaliza el texto de tal forma que se facilite el emparejamiento de palabras u oraciones [63].

6.1.3 Eliminación de palabras vacías

Las "palabras vacías" son términos considerados como ruido debido a su bajo contenido semántico, generalmente son términos que se encuentra con una alta frecuencia dentro del texto y no aportan a la tarea de distinguir las oraciones más importantes del texto. En este proyecto esta tarea se realizó mediante la aplicación de un filtro de términos usando la lista de palabras vacías construida para el sistema de recuperación de información *SMART*[8] [89].

6.1.4 Lematización

Por razones gramaticales, los documentos van a utilizar diferentes formas de una palabra, debido a que existen familias de palabras derivadas con significados similares, como por ejemplo "democracia", "democrático", y "democratización" [68]. En este sentido el objetivo de la lematización es reducir las formas de inflexión y palabras derivadas en una base común o misma raíz, para disminuir el impacto de la diferencia entre conceptos semejantes que contribuye a reducir la cantidad de recursos de almacenamiento, tiempo de ejecución y el tamaño de la estructura de indexación [90]. En la presente investigación, el algoritmo utilizado es el de Porter[9] [91] que realiza una eliminación de sufijos y posteriormente recodifican la cadena de texto tratada.

6.1.5 Indexación

En el proceso de indexación de texto las oraciones recolectadas y normalizadas son almacenadas en una estructura de datos para facilitar el acceso rápido y exacto por parte del proceso de recuperación de información, mediante el cual se realiza la búsqueda y emparejamiento de términos u oraciones [92]. En el presente trabajo, la librería de Lucene[10] es utilizada para la indexación de términos, a la vez que contribuye a las tareas de eliminación de mayúsculas y signos ortográficos, eliminación de palabras vacías y lematización. Lucene es una librería de código abierto bajo la licencia Apache Software Licence, cuyo objetivo es facilitar la indexación y búsqueda en tareas de recuperación de información. En la actualidad está disponible en diversos lenguajes de programación como Java, C#, C++, Delphi, PHP, Phyton y Ruby [93]. Una de las características principales de esta herramienta, es la abstracción de los documentos como un conjunto de campos de texto, muy útil para el acoplamiento con sistemas basados en el Modelo de Espacio Vectorial para la representación de los documentos.

[8] Esta lista se encuentra disponible en *ftp://ftp.cs.cornell.edu/pub/smart/english.stop*.
[9] Disponible en http://tartarus.org/martin/PorterStemmer/
[10] Esta librería se encuentra disponible en http://lucenenet.apache.org/

6.2 COLECCIÓN DE DOCUMENTOS DE EVALUACIÓN

La Conferencia de Entendimiento del Documento (DUC, Document Understanding Conference) define resúmenes "ideales" creados por humanos expertos sobre un conjunto de documentos originales, con el objetivo de ofrecer a la comunidad académica conjuntos de datos que permitan evaluar y comparar los resultados obtenidos por sistemas de generación automática de resúmenes. Para evaluar el algoritmo FSP-MT-SingleDocSum, se utilizan específicamente los conjuntos de datos DUC2001 y DUC2002, que contienen artículos de noticias en inglés. DUC2001 es una colección de 309 artículos distribuidos en 30 tópicos, mientras que DUC2002 es una colección de 567 artículos con 59 tópicos. En las dos colecciones, cada artículo viene acompañado por un resumen de referencia con una longitud de 100 palabras (Ver Tabla 28).

	DUC2002	DUC2001
Número de conjuntos	59	30
Número de documentos	567	309
Fuente de datos	TREC[11]	TREC
Longitud del resumen	100 palabras	100 palabras

Tabla 28. Resumen de los conjuntos de datos utilizados

6.3 MÉTRICAS DE EVALUACIÓN

La evaluación de la calidad de los resúmenes generados por el algoritmo FSP-MT-SingleDocSum fue realizada usando el conjunto de métricas provenientes de la herramienta de evaluación ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [94] en su versión 1.5.5. Esta herramienta permite comparar el contenido de un resumen con uno o más resúmenes de referencia, y obtener el número de n-gramas de palabras que tienen en común. En esta investigación se usaron las medidas ROUGE-N (con N=1 y N=2) y ROUGE SU4 debido a que son las más usadas por los métodos del estado del arte para comparar los resultados de sistemas de generación automática de resúmenes de un documento.

6.4 AFINACION DE PARAMETROS

Los valores de los parámetros del algoritmo FSP-MT-SingleDocSum fueron obtenidos luego de realizar un proceso de afinación de los parámetros del algoritmo FSP, asociados al problema y de memoria tabú.

Los primeros parámetros en ser afinados fueron los del algoritmo FSP, por medio de combinaciones de T, N, L y M, en las cuales el producto de estos parámetros, no supera el total de evaluaciones de la función objetivo establecido en 1600 (Para realizar la comparación con los métodos del estado del arte). Para el parámetro C se definieron dos

[11] http://trec.nist.gov/overview.html

esquemas: *C* estático y *C* dinámico. El mejor resultado lo obtuvo *C Estático* (ver sección 4.3.3).

La afinación de los parámetros asociados al problema utilizo la configuración preliminar de la Tabla 2. Este proceso afino cada parámetro en el siguiente orden: el primero *Máxima longitud a evaluar de un resumen*, tomando los valores 100, 110, 120 y 130, y además se realizó una afinación de los parámetros FSP para cada uno de esos valores, el mejor resultado se obtuvo con el valor 130 y la combinación de los parámetros T, N, L y M con los valores 6, 13, 2, 10 respectivamente; el segundo *Criterio para deshabilitar una oración*, se tomó cada una de las características de la función objetivo y algunas combinaciones de ellas, el mejor resultado se obtuvo con la combinación de características Posición-Cobertura; el tercero *Criterio de Selección de Oraciones al generar el resumen*, también se tomó cada una de las características de la función objetivo y algunas combinaciones de las características que mejores resultados mostraron en el anterior criterio, el mejor resultado se obtuvo con la característica Posición. El parámetro *Máximo Número de Evaluaciones de la función objetivo* no necesito afinación debido a que su valor se estableció en 1600 para cumplir una de las restricciones de comparación del algoritmo propuesto con otros del estado del arte.

Por último la afinación de los parámetros de la memoria tabú, utilizo las configuraciones de los parámetros FSP y asociados al problema presentadas en las Tabla 10 y Tabla 13, respectivamente. En este caso se experimentó con los cuatro tipos de memoria tabú: Explícita Global, Explícita por Punto de Captura, Implícita por Movimientos e Implícita por Atributos. Cada memoria tomó varios valores del parámetro $tenure$. El mejor resultado se obtuvo con la Memoria Tabú Explícita por Punto de Captura y el parámetro $tenure$ con valor 60.

La configuración final de los parámetros de FSP-MT-SingleDocSum, se obtuvo al finalizar los ciclos de experimentación y se presenta en la Tabla 29.

	Parámetro	Valor
Parámetros FSP	T	6
	N	13
	L	2
	M	10
	C Estático	1
Parámetros Asociados al Problema	Máximo Número de Evaluaciones de la función objetivo	1600
	Máxima Longitud a Evaluar de un resumen	130
	Criterio para deshabilitar una oración	Posición y Cobertura
	Criterio de Selección de Oraciones al generar el resumen	Posición
Parámetros Memoria Tabú	Tenure	60
	Tipo de Memoria Tabú	Explícita por Punto de Captura

Tabla 29. Parámetros Finales FSP-MT-SingleDocSum.

FSP-MT-SingleDocSum fue ejecutado treinta veces por documento, evaluando los resúmenes generados en cada ejecución a través de las métricas de ROUGE, obteniendo al final un resultado promedio de todo el conjunto de documentos por cada métrica. La implementación del algoritmo se realizó en el lenguaje de programación C# de la plataforma .NET, las ejecuciones del algoritmo se realizaron en un computador de escritorio con procesador Intel Core i5 3.2 GHz, RAM de 8 GB, sistema operativo Windows 7.

6.5 COMPARACIÓN CON OTROS MÉTODOS DEL ESTADO DEL ARTE

Los resultados obtenidos por FSP-MT-SingleDocSum son comparados con los siguientes métodos del estado del arte de generación automática de resúmenes de un documento, basados en: aprendizaje de maquina NetSum [22], CRF [23] y QCS [61]; reducción algebraica SVM [15]; grafos UnifiedRank [33] y Manifold Ranking [95]; metaheurísticas FEOM [16], MA-SingleDocSum [51] y DE [52].

Los resultados de las medidas ROUGE obtenidas por el algoritmo FSP-MT-SingleDocSum y otros métodos del estado del arte para los conjuntos de datos DUC2001 son presentados en la Tabla 30; y en la Tabla 31 los resultados para DUC2002.

De acuerdo a los datos presentados en las Tabla 30 y Tabla 31, se puede observar que FSP-MT-SingleDocSum en la medida ROUGE-2 supera a todos los métodos del estado del arte sobre los conjuntos de datos de DUC2001 y DUC2002; en la medida ROUGE-1 para DUC2002, FSP-MT-SingleDocSum es primero y en DUC2001 se ubica en cuarto lugar.

Método	ROUGE-1		ROUGE-2	
FSP-MT-SingleDocSum	0,46004	4	**0,20582**	1
MA-SingleDocSum	0.44862	7	0.20142	2
DE	**0.47856**	1	0.18528	4
UnifiedRank	0.45377	6	0.17646	7
FEOM	0.47728	2	0.18549	3
NetSum	0.46427	3	0.17697	6
CRF	0.45512	5	0.17327	8
QSC	0.44852	8	0.18523	5
SVM	0.44628	9	0.17018	9
Manifold Ranking	0.43359	10	0.16635	10

Tabla 30. Puntajes ROUGE de los métodos sobre DUC2001.

Método	ROUGE-1		ROUGE-2	
FSP-MT-SingleDocSum	**0,48862**	1	**0,23056**	1
MA-SingleDocSum	0.48280	3	0.22840	2
DE	0.46694	4	0,12368	6
UnifiedRank	0.48487	2	0,21462	3
FEOM	0.46575	5	0,12490	5
NetSum	0.44963	6	0.11167	7
CRF	0.44006	8	0.10924	8
QSC	0.44865	7	0.18766	4
SVM	0.43235	10	0.10867	9
Manifold Ranking	0.42325	9	0.10677	10

Tabla 31. Puntajes ROUGE de los métodos sobre DUC2002.

Para calcular el porcentaje de mejora de los resultados obtenidos por FSP-MT-SingleDocSum con respecto a los otros métodos, se utiliza la Ecuación (6.1).

$$\frac{MetodoPropuesto - OtroMetodo}{OtroMetodo} \times 100 \qquad (6.1)$$

En la Tabla 32 se presentan estos porcentajes en las medidas ROUGE-1 para DUC2002 y ROUGE-2 para DUC2001 y DUC2002. Con respecto a ROUGE-1 con DUC2002, FSP-MT-SingleDocSum supera a UnifiedRank, MA-SingleDocSum y DE con porcentajes de 0.77%, 1.21% y 4.64% respectivamente. De otro lado respecto a ROUGE-2 con DUC2001 mejora sobre MA-SingleDocSum, DE y FEOM por 2.18%, 11.09% y 10.96% respectivamente, y con DUC2002, FSP-MT-SingleDocSum mejora sobre MA-SingleDocSum, UnifiedRank y QSC con porcentajes de 0.95%, 7.43% y 22.86% respectivamente.

Método	% Mejora obtenida por FSP-MT-SingleDocSum (%)		
	ROUGE-1	ROUGE-2	
	DUC2002	DUC2001	DUC2002
MA-SingleDocSum	1,21	**2,18**	**0,95**
DE	4,64	11,09	86,42
UnifiedRank	**0,77**	16,64	7,43
FEOM	4,91	10,96	84,60
NetSum	8,67	16,30	106,47
CRF	11,03	18,79	111,06
QSC	8,91	11,12	22,86
SVM	13,01	20,94	112,17
Manifold Ranking	15,44	23,73	115,94

Tabla 32. Comparación de FSP-MT-SingleDocSum con otros métodos.

El porcentaje de mejora obtenido por DE en la medida ROUGE-1 sobre DUC2001 respecto a los otros métodos, se presenta en la Tabla 33. Respecto a ROUGE-1 con DUC2001 DE supera a FEOM, Netsum, FSP-SingleDocSum con porcentajes de 0.27%, 3.08% y 4.03% respectivamente.

Método	Mejora obtenida por DE (%)
	ROUGE-1
	DUC2001
FSP-MT-SingleDocSum	4.03
MA-SingleDocSum	6.67
Unified Rank	5.46
FEOM	**0.27**
NetSum	3.08
CRF	5.15
QSC	6.70
SVM	7.23
Manifold Ranking	10.37

Tabla 33. Comparación de DE con otros métodos.

Debido a que los resultados no identifican que método obtiene los mejores resultados en ambos conjuntos de datos, se utiliza un ranking unificado de todos los métodos, que toma en cuenta la posición que ocupa cada método en cada medida, de acuerdo a la fórmula de la Ecuación (6.2) extraída de [96].

$$Ran(Metodo) = \sum_{r=1}^{10} \frac{(10 - r + 1) \times R_r}{10} \tag{6.2}$$

Donde R_r indica el número de veces que el método aparece clasificado en la posición r. El número diez representa el número total de métodos que están siendo comparados.

Considerando los resultados de la Tabla 34, se observa que el algoritmo FSP-MT-SingleDocSum se ubica de primero en este ranking, con tres primeros lugares y un cuarto lugar, superando a los otros métodos del estado del arte. También que los algoritmos basados en metaheurísticas: FSP-MT-SingleDocSum, MA-SingleDocSum, DE y FEOM, que tratan el problema de la generación automática de resúmenes como un problema de optimización superan a los otros métodos del estado del arte.

Método	R_r										Clasificación Resultante
	1	2	3	4	5	6	7	8	9	10	
FSP-MT-SingleDocSum	3	0	0	1	0	0	0	0	0	0	**3,7**
MA-SingleDocSum	0	2	1	0	0	0	1	0	0	0	3
DE	1	0	0	2	0	1	0	0	0	0	2,9
FEOM	0	1	1	0	2	0	0	0	0	0	2,9
UnifiedRank	0	1	1	0	0	1	1	0	0	0	2,6
NetSum	0	0	1	0	0	2	1	0	0	0	2,2
QSC	0	0	0	1	1	0	1	1	0	0	2
CRF	0	0	0	0	1	0	0	3	0	0	1,5
SVM	0	0	0	0	0	0	0	0	3	1	0,7
Manifold Ranking	0	0	0	0	0	0	0	0	1	3	0,5

Tabla 34. Ranking de Clasificación de los algoritmos.

6.6 COMPARACIONES ADICIONALES

Con el propósito de analizar los resultados obtenidos al adaptar el algoritmo FSP con los algoritmos Ascenso de Colina y Temple Simulado y una memoria tabú, se presentan en la Tabla 35 los mejores resultados obtenidos por los siguientes algoritmos:

- FSP-SingleDocSum: Adaptación de FSP a la generación automática de resúmenes, obtenido al finalizar el ciclo I (ver sección 3.1).
- FSP-ASC-SingleDocSum: Adaptación de FSP con el optimizador local Ascenso de la Colina a la generación automática de resúmenes, obtenido al finalizar el ciclo II (ver sección 3.2).
- FSP-TS-SingleDocSum: Adaptación de FSP con el optimizador local Temple Simulado a la generación automática de resúmenes, obtenido al finalizar el ciclo III (ver sección 3.3).
- FSP-MT-SingleDocSum: Adaptación de FSP con una memoria tabú a la generación automática de resúmenes, obtenido al finalizar el ciclo IV (ver sección 3.4).

Algoritmo	**DUC2001**			**DUC2002**		
	R1R	R2R	RSU4R	R1R	R2R	RSU4R
FSP-SingleDocSum	0,45982	0,20589	0,22852	0,48809	0,23015	0,24666
FSP-ASC-SingleDocSum	**0,46027**	**0,20627**	0,22784	0,48801	0,22992	0,24645
FSP-TS-SingleDocSum	0,4597	0,20604	0,22791	0,48799	0,22992	0,24639
FSP-MT-SingleDocSum	0,46004	0,20582	**0,22845**	**0,48862**	**0,23056**	**0,24705**

Tabla 35. Resultados de la adaptación estrategias de búsqueda local.

En la Tabla 36 se muestra el mejoramiento relativo, calculado como en la Ecuación (6.1), de los algoritmos FSP-ASC-SingleDocSum, FSP-TS-SingleDocSum y FSP-MT-SingleDocSum, respecto al algoritmo FSP-SingleDocSum. Donde se observa el porcentaje que creció o decreció cada una de las medidas ROUGE obtenidas por los algoritmos que adaptan una estrategia de búsqueda local o memoria tabú frente al algoritmo que no las utiliza.

Algoritmo	DUC2001			DUC2002		
	R1R %	R2R %	RSU4R %	R1R %	R2R %	RSU4R %
FSP-ASC-SingleDocSum	**0,10**	**0,18**	-0,30	-0,02	-0,10	-0,09
FSP-TS-SingleDocSum	-0,03	0,07	-0,27	-0,02	-0,10	-0,11
FSP-MT-SingleDocSum	0,05	-0,03	-0,03	**0,11**	**0,18**	**0,16**

Tabla 36. Mejoramiento relativo al aplicar estrategias de búsqueda local

Conforme a los resultados presentados en la Tabla 36 se observa que:

- El algoritmo FSP-ASC-SingleDocSum muestra un menor desempeño respecto al algoritmo FSP-SingleDocSum, ya que disminuye el valor en cuatro medidas (ROUGE-SU4 Recall sobre DUC2001 y ROUGE-1 Recall, ROUGE-2 Recall y ROUGE-SU4 Recall sobre DUC2002) y solo obtiene mejores resultados en dos medidas (ROUGE-1 Recall y ROUGE-2 Recall sobre DUC2001).

- El algoritmo FSP-TS-SingleDocSum muestra un menor desempeño, debido a que solo obtuvo mejor resultado en las medida ROUGE-2 Recall sobre DUC2001 y presento resultados más bajos en las demás medidas.

- El algoritmo FSP-MT-SingleDocSum muestra un mejor desempeño, ya que obtuvo mejores resultados en cuatro medidas (ROUGE-1 Recall sobre DUC2001 y ROUGE-1 Recall, ROUGE-2 Recall, ROUGE-SU4 Recall sobre DUC2002) y solo en dos medidas (ROUGE-2 Recall y ROUGE-SU4 Recall sobre DUC2001) disminuyó el valor en un bajo porcentaje.

- El menor desempeño de los algoritmos FSP-ASC-SingleDocSum y FSP-TS-SingleDocSum respecto a FSP-SingleDocSum se debe a que el número máximo de evaluaciones de la función objetivo, limitó el número de optimizaciones de los optimizadores para encontrar mejores soluciones.

- El algoritmo FSP-MT-SingleDocSum muestra un mejor desempeño respecto al algoritmo FSP-SingleDocSum, debido a que con la adaptación de una memoria tabú explícita permite mejorar la fase de explotación de cada punto de captura, ya que al mantener un porcentaje de las soluciones vecinas como tabú, se intensifica la búsqueda de nuevas soluciones en la vecindad de cada punto de captura.

Capítulo 7

7 CONCLUSIONES Y TRABAJO FUTURO

7.1 CONCLUSIONES

En este trabajo el algoritmo FSP fue adaptado para resolver el problema de la generación automática de resúmenes extractivos de un documento como un problema de optimización binaria. Las modificaciones necesarias para llevar a cabo esta adaptación incluyeron: la selección de la representación de soluciones binaria por su simplicidad y flexibilidad para codificar y trabajar con problemas de múltiples variables y la facilidad para ser decodificada en la solución real del problema; el parámetro coeficiente de amplitud (C) se adaptó para que su valor indique el número de oraciones a ajustar en una solución; la creación del método Generar Vecino que añade conocimiento específico del problema a la creación de nuevas soluciones candidatas a través del parámetro *Criterio para Deshabilitar un Oración* que estableció las características Posición y Cobertura para puntuar las oraciones y luego escoger la de menor puntaje para ser deshabilitada, y con el procedimiento *Reparar Solución* que ajusto las oraciones para cumplan el límite de palabras del resumen; por último la implementación de una memoria tabú a través de listas que almacenan las soluciones visitadas.

La adaptación de una memoria tabú explícita permite mejorar la fase de explotación de cada punto de captura, ya que al mantener un porcentaje de las soluciones vecinas como tabú, se intensifica la búsqueda de nuevas soluciones en la vecindad de cada punto de captura. y evita un uso innecesario de evaluaciones de la función objetivo en soluciones ya evaluadas.

La función objetivo definida está basada en la función objetivo propuesta en [51] que permite evaluar las características de las oraciones que componen un resumen candidato como: *Posición*, *Relación con el Título* y *Longitud*, y otras características que miden la relación entre las oraciones del resumen y el grado en que las oraciones del resumen presentan la información más importante del documento, *Cohesión y Cobertura* respectivamente. La función objetivo propuesta en esta investigación presenta cambios totales o parciales en la forma de calcular algunas características con respecto a la función objetivo presentada en [51] para conseguir que los valores obtenidos por las características se encuentren en el mismo rango. La posición fue calculada para permitir a diferencia de la anterior ecuación una evaluación más balanceada de las oraciones según su posición, debido a que el puntaje asignado a cada oración va disminuyendo en la misma proporción desde la primera hasta la última oración. La ecuación de la longitud se modificó al dividirla entre el número de oraciones del resumen, para mejorar los puntajes de los resúmenes con menos oraciones cortas y acotar los puntajes obtenidos por esta característica a valores menores que uno. La ecuación de la cohesión se modificó debido a que en esta investigación la matriz de similitud de oraciones se encuentra normalizada y por lo tanto la función logaritmo utilizada para ajustar los valores de esta ecuación no era necesaria. La cobertura fue calculada con una nueva ecuación que a diferencia de la

anterior calcula directamente la similitud de cosenos entre el vector de términos que representa al resumen con el vector de términos que representa al documento. Los cambios aplicados a estas características mostraron ser efectivos debido a que los resultados obtenidos por FSP-MT-SingleDocSum superaron a los resultados de MA-SingleDocSum y a los otros métodos del estado del arte.

Los algoritmos FSP-SingleDocSum y FSP-MT-SingleDocSum obtuvieron mejores resultados que los algoritmos FSP-ASC-SingleDocSum y FSP-TS-SingleDocSum, debido a que estos últimos utilizan un número restringido de evaluaciones de la función objetivo que no les permite realizar una explotación aceptable de los puntos de captura.

La calidad de los resúmenes generados automáticamente por el algoritmo FSP-MT-SingleDocSum fue evaluada por medio de las medidas ROUGE-1, ROUGE-2 y ROUGE-SU4 sobre los conjuntos de datos DUC2001 y DUC2002. Al comparar los resultados obtenidos se observó que el algoritmo propuesto supera a otros algoritmos metaheurísticos del estado del arte, con porcentajes de mejora sobre el mejor método (MA-SingleDocSum) de 1,21% en ROUGE-1 sobre DUC2002, de 2,18% y 0,95% en ROUGE-2 sobre DUC2001 y DUC2002 respectivamente. Por otra parte, comparado con otros métodos no metaheurísticos, FSP-MT-SingleDocSum supera al mejor método (UnifiedRank) en las medidas ROUGE-1 en un 0,77% para DUC2002 y ROUGE-2 en un 7,43%, para DUC2002 y al mejor método (QSC) en la medida ROUGE-2 en un 11,12% para DUC2001. Estos resultados indican un buen desempeño del FSP-MT-SingleDocSum frente a los trabajos del estado del arte, aunque en la medida de ROUGE-1 sobre DUC2001 el método de evolución diferencial supera a FSP-MT-SingleDocSum en 4.03%, FSP-MT-SingleDocSum obtiene los mejores resultados en las otras medidas. Además se realizó una comparación utilizando un método unificado que clasifico a los algoritmos según las posiciones que ocuparon en las medidas ROUGE-1 y ROUGE-2 sobre DUC2001 y DUC2002, donde FSP-MT-SingleDocSum obtuvo el primer lugar de la clasificación con tres primeros lugares (ROUGE-1 en DUC2002, ROUGE-2 en DUC2001 y DUC2002) y un cuarto lugar (ROUGE-1 sobre DUC2001), superando a los otros métodos del estado del arte.

En este trabajo se diseñó, desarrolló y liberó el complemento FSP-MT-Summarizer para el programa ofimático Office Word que permite generar resúmenes extractivos automáticos de un documento de texto utilizando el algoritmo FSP-MT-SingleDocSum.

7.2 RECOMENDACIONES Y TRABAJO FUTURO

Teniendo en cuenta los resultados obtenidos con el algoritmo FSP-MT-SingleDocSum, se espera estudiar otros esquemas de inicialización, ajuste y reemplazo, otras fórmulas para el cálculo de las similitudes, diferentes formas de representación y ponderación de los términos, entre otros aspectos, con el objetivo de intentar mejorar los resultados obtenidos por el algoritmo propuesto.

Llevar a cabo un estudio del comportamiento de la función objetivo propuesta en este proyecto con diferentes tipos de documentos (que no sean noticias), para determinar si es aplicable directamente sobre documentos de diversos géneros o si es necesaria su adaptación.

Evaluar el comportamiento de FSP-MT-SingleDocSum con otros conjuntos de datos de DUC, para intentar generalizar las conclusiones obtenidas del desempeño del FSP-MT-SingleDocSum en esta investigación.

BIBLIOGRAFÍA

[1] N. Kumaresh and B. Ramakrishnan, "Graph Based Single Document Summarization," in *Data Engineering and Management*. vol. 6411, R. Kannan and F. Andres, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 32-35.

[2] A. Porselvi and S. Gunasundari, "Survey on web page visual summarization," *International Journal of Emerging Technology and Advanced Engineering,* vol. 3, pp. 26-32, 2013.

[3] A. Alhindi, U. Kruschwitz, C. Fox, and M.-D. Albakour, "Profile-Based Summarisation for Web Site Navigation," *ACM Trans. Inf. Syst.,* vol. 33, pp. 1-39, 2015.

[4] D. M. Zajic, B. J. Dorr, and J. Lin, "Single-document and multi-document summarization techniques for email threads using sentence compression," *Information Processing & Management,* vol. 44, pp. 1600-1610, 2008.

[5] H. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research and Development,* vol. 2, pp. 159-165, 1958.

[6] A. Nenkova and K. McKeown, "A Survey of Text Summarization Techniques," in *Mining Text Data*, C. C. Aggarwal and C. Zhai, Eds., ed: Springer US, 2012, pp. 43-76.

[7] E. Lloret and M. Palomar, "Text summarisation in progress: a literature review," *Artificial Intelligence Review,* vol. 37, pp. 1-41, 2012/01/01 2012.

[8] M. E. MENDOZA BECERRA and E. LEON GUZMÁN, *Una Revisión de la Generación Automática de Resúmenes Extractivos* vol. 12, 2013.

[9] H. P. Edmundson, "New Methods in Automatic Extracting," *Journal of the ACM (JACM),* vol. 16, pp. 264-285, 1969.

[10] Y. K. Meena and D. Gopalani, "Feature Priority Based Sentence Filtering Method for Extractive Automatic Text Summarization," *Procedia Computer Science,* vol. 48, pp. 728-734, // 2015.

[11] Y. Gong and X. Liu, "Generic text summarization using relevance measure and latentsemantic analysis," in *24th annual international ACM SIGIR conference on Research and development in information retrieval*, New Orleans, USA, 2001, pp. pp.19-25.

[12] D. Tsarev, M. Petrovskiy, and I. Mashechkin, "Using NMF-based text summarization to improve supervised and unsupervised classification," in *Hybrid Intelligent Systems (HIS), 2011 11th International Conference on*, 2011, pp. 185-189.

[13] M. G. Ozsoy, F. N. Alpaslan, and I. Cicekli, "Text summarization using Latent Semantic Analysis," *J. Inf. Sci.,* vol. 37, pp. 405-417, 2011.

[14] J. Steinberger and K. Ježek, "Using latent semantic analysis in text summarization and summary evaluation," in *7th International Conference ISIM*, 2004.

[15] J.-Y. Yeh, H.-R. Ke, W.-P. Yang, and I.-H. Meng, "Text summarization using a trainable summarizer and latent semantic analysis," *Information Processing and Management,* vol. 41, pp. 75–95, 2005.

[16] J. Steinberger and K. Ježek, "Sentence Compression for the LSA-based Summarizer," pp. 141–148, 2006.

[17] J.-H. Lee, S. Park, C.-M. Ahn, and D. Kim, "Automatic generic document summarization based on non-negative matrix factorization," *Information Processing & Management,* vol. 45, pp. 20-34, 2009.

[18] J. Kupiec, J. Pedersen, and F. Chen, "A trainable document summarizer," in *18th annual international ACM SIGIR conference on Research and development in information retrieval*, Seattle, Washington, United States, pp. 68-73, 1995.

[19] M. E. Hannah, T. V. Geetha, and S. Mukherjee, "Automatic extractive text summarization based on fuzzy logic: a sentence oriented approach," presented at the Proceedings of the Second international conference on Swarm, Evolutionary, and Memetic Computing - Volume Part I, Visakhapatnam, Andhra Pradesh, India, 2011.

[20] F. Kyoomarsi, H. Khosravi, E. Eslami, P. K. Dehkordy, and A. Tajoddin, "Optimizing Text Summarization Based on Fuzzy Logic," in *Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on*, 2008, pp. 347-352.

[21] C. Aone, M. E. Okurowski, J. Gorlinsky, and B. s. Larsen, "A trainable summarizer with knowledge acquired from robust NPL techniques," *Advances in Automatic Text Summarization,* vol. Mani, I. and Maybury, M. T., pp. 71-80, 1999.

[22] K. Svore, L. Vanderwende, and C. Burges, "Enhancing single-document summarization by combining RankNet and third-party sources," in *Empirical Methods in Natural Language Processing (EMNLP-CoNLL)*, 2007, pp. 448-457.

[23] D. Shen, J.-T. Sun, H. Li, Q. Yang, and Z. Chen, "Document summarization using conditional random fields," in *20th international joint conference on Artifical intelligence*, Hyderabad, India, 2007, pp. 2862-2867.

[24] K.-F. Wong, M. Wu, and W. Li, "Extractive summarization using supervised and semi-supervised learning," in *22nd International Conference on Computational Linguistics*, Manchester, United Kingdom, 2008, pp. 985-992.

[25] J. Conroy and D. O'leary, "Text summarization via hidden Markov models," in *24th annual international ACM SIGIR conference on Research and development in information retrieval*, New Orleans, Louisiana, United States, 2001, pp. 406-407.

[26] D. Muratore, M. Hagenbuchner, F. Scarselli, and A. C. Tsoi, "Sentence extraction by graph neural networks," presented at the Proceedings of the 20th international conference on Artificial neural networks: Part III, Thessaloniki, Greece, 2010.

[27] D. Marcu, " Improving summarization through rhetorical parsing tuning," in *Sixth Workshop on Very Large Corpora. Montreal, Canada*, 1998, pp. 206-215.

[28] A. Ibrahim and T. Elghazaly, "Improve the Automatic Summarization of Arabic Text Depending on Rhetorical Structure Theory," presented at the Proceedings of the 2013 12th Mexican International Conference on Artificial Intelligence, 2013.

[29] R. Barzilay, Elhadad, M, "Using Lexical Chains for Text Summarization," in *ACL/EACL Workshop on Intelligent Scalable Text Summarization, Madrid, Spain*, 1997, pp. 10–17.

[30] A. Louis, A. Joshi, and A. Nenkova, "Discourse indicators for content selection in summarization," presented at the Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Tokyo, Japan, 2010.

[31] K. Ono, K. Sumita, and S. Miike, "Abstract generation based on rhetorical structure extraction," in *15th conference on Computational linguistics*, Kyoto, Japan, 1994, pp. 344-348.

[32] A. R. Pal and D. Saha, "An approach to automatic text summarization using WordNet," in *Advance Computing Conference (IACC), 2014 IEEE International*, 2014, pp. 1169-1173.

[33] X. Wan, "Towards a unified approach to simultaneous single-document and multi-document summarizations," presented at the Proceedings of the 23rd International Conference on Computational Linguistics, Beijing, China, 2010.

[34] D. R. Amancio, M. G. V. Nunes, O. N. Oliveira Jr, and L. d. F. Costa, "Extractive summarization using complex networks and syntactic dependency," *Physica A: Statistical Mechanics and its Applications,* vol. 391, pp. 1855-1864, 2/15/ 2012.

[35] R. Mihalcea, Tarau, P, "Text-rank: bringing order into texts," in *Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain*, 2004.

[36] Y. Ledeneva, R. García-Hernández, and A. Gelbukh, "Graph Ranking on Maximal Frequent Sequences for Single Extractive Text Summarization," in *Computational Linguistics and Intelligent Text Processing*. vol. 8404, A. Gelbukh, Ed., ed: Springer Berlin Heidelberg, 2014, pp. 466-480.

[37] F. Rafael, "A Four Dimension Graph Model for Automatic Text Summarization," 2013, pp. 389-396.

[38] N. Chatterjee and P. K. Sahoo, "Random Indexing and Modified Random Indexing based approach for extractive text summarization," *Computer Speech & Language,* vol. 29, pp. 32-44, 1// 2015.

[39] Y. Ledeneva, R. Hernández, R. Soto, R. Reyes, and A. Gelbukh, "EM Clustering Algorithm for Automatic Text Summarization," in *Advances in Artificial Intelligence*. vol. 7094, I. Batyrshin and G. Sidorov, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 305-315.

[40] R. Yazhini and R. P. Vishnu, "Automatic summarizer for mobile devices using sentence ranking measure," in *Recent Trends in Information Technology (ICRTIT), 2014 International Conference on*, 2014, pp. 1-6.

[41] M. A. Ramiz, "A Novel Partitioning-Based Clustering Method and Generic Document Summarization," in *Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*, 2006.

[42] E. Shareghi and L. S. Hassanabadi, "Text summarization with harmony search algorithm-based sentence extraction," in *5th international conference on Soft computing as transdisciplinary science and technology* Cergy-Pontoise, France, 2008, pp. 226-231.

[43] M. S. Binwahlan, N. Salim, and L. Suanmali, "Swarm Based Text Summarization," in *International Association of Computer Science and Information Technology - Spring Conference. (IACSITSC)*, Singapore, 2009, pp. 145-150.

[44] H. Asgari, B. Masoumi, and O. S. Sheijani, "Automatic text summarization based on multi-agent particle swarm optimization," in *Intelligent Systems (ICIS), 2014 Iranian Conference on*, 2014, pp. 1-5.

[45] V. Qazvinian, L. Sharif, and R. Halavati, "Summarising text with a genetic algorithm-based sentence extraction," *International Journal of Knowledge Management Studies (IJKMS),* vol. 4, pp. 426-444, 2008.

[46] M. Litvak, M. Last, and M. Friedman, "A new approach to improving multilingual summarization using a genetic algorithm," in *48th Annual Meeting of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden, 2010, pp. 927-936.

[47] R. García-Hernández and Y. Ledeneva, "Single Extractive Text Summarization Based on a Genetic Algorithm," in *Pattern Recognition*. vol. 7914, J. Carrasco-Ochoa, J. Martínez-Trinidad, J. Rodríguez, and G. Baja, Eds., ed: Springer Berlin Heidelberg, 2013, pp. 374-383.

[48] N. Chatterjee, A. Mittal, and S. Goyal, "Single document extractive text summarization using Genetic Algorithms," in *Emerging Applications of Information Technology (EAIT), 2012 Third International Conference on*, 2012, pp. 19-23.

[49] P.-K. Dehkordi, F. Kumarci, and H. Khosravi, "Text Summarization Based on Genetic Programming," *International Journal of Computing and ICT Research,* vol. 3, pp. 57-64, 2009.

[50] U. Nguyen Quang, "A Study on the Use of Genetic Programming for Automatic Text Summarization," 2012, pp. 93-98.

[51] M. Mendoza, S. Bonilla, C. Noguera, C. Cobos, and E. León, "Extractive single-document summarization based on genetic operators and guided local search," *Expert Systems with Applications,* vol. 41, pp. 4158-4169, 2014.

[52] R. M. Aliguliyev, "A new sentence similarity measure and sentence based extractive technique for automatic text summarization," *Expert Systems with Applications,* vol. 36, pp. 7764-7772, 2009.

[53] A. Abuobieda, N. Salim, Y. Kumar, and A. Osman, "An Improved Evolutionary Algorithm for Extractive Text Summarization," in *Intelligent Information and Database Systems*. vol. 7803, A. Selamat, N. Nguyen, and H. Haron, Eds., ed: Springer Berlin Heidelberg, 2013, pp. 78-89.

[54] W. Song, L. Cheon Choi, S. Cheol Park, and X. Feng Ding, "Fuzzy evolutionary optimization modeling and its applications to unsupervised categorization and extractive summarization," *Expert Systems with Applications,* vol. 38, pp. 9112-9121, 2011.

[55] M. S. Binwahlan, N. Salim, and L. Suanmali, "Fuzzy swarm diversity hybrid model for text summarization," *Information Processing & Management,* vol. 46, pp. 571-588, 2010.

[56] R. A. Ghalehtaki, H. Khotanlou, and M. Esmaeilpour, "A combinational method of fuzzy, particle swarm optimization and cellular learning automata for text summarization," in *Intelligent Systems (ICIS), 2014 Iranian Conference on*, 2014, pp. 1-6.

[57] L. Suanmali, N. Salim, and M. S. Binwahlan, "Fuzzy Genetic Semantic Based Text Summarization," presented at the Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, 2011.

[58] E. González and M. Fuentes, "A New Lexical Chain Algorithm Used for Automatic Summarization," presented at the Proceedings of the 2009 conference on Artificial Intelligence Research and Development: Proceedings of the 12th International Conference of the Catalan Association for Artificial Intelligence, 2009.

[59] O. Alejo-Machado, J. Fernández-Luna, J. Huete, and E. C. Morales, "Fisherman search procedure," *Progress in Artificial Intelligence,* vol. 2, pp. 193-203, 2014/07/01 2014.

[60] K. S. Pratt, "Design Patterns for Research Methods: Iterative Field Research," in *Association for the Advancement of Artificial Intelligence*, 2009.

[61] D. M. Dunlavy, D. P. O'Leary, J. M. Conroy, and J. D. Schlesinger, "QCS: A system for querying, clustering and summarizing documents," *Information Processing & Management,* vol. 43, pp. 1588-1605, 2007.

[62] A. Abuobieda, N. Salim, Y. Kumar, and A. Osman, "Opposition Differential Evolution Based Method for Text Summarization," in *Intelligent Information and Database Systems*. vol. 7802, A. Selamat, N. Nguyen, and H. Haron, Eds., ed: Springer Berlin Heidelberg, 2013, pp. 487-496.

[63] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*: Cambridge University Press, 2008.

[64] M. Hassel, "Resource Lean and Portable Automatic Text Summarization," Doctoral, Computer Science and Communication, KTH School of Computer Science and Communication, Stockholm, Sweden 2007.

[65] C. J. V. Rijsbergen, *Information Retrieval*. Newton, MA, USA: Butterworth-Heinemann, 1979.

[66] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop,* pp. 74-81, July 2004.

[67] G. Salton, A. Wong, and C. Yang, "A vector space model for information retrieval," *Communications of The ACM,* 1975.

[68] A. Singhal, "Modern Information Retrieval: A Brief Overview," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering,* vol. 24, pp. 35-42, 2001.

[69] E. Villatoro Tello, "Generación automática de resúmenes de múltiples documentos," Instituto Nacional de Astrofísica, Óptica y Electrónica, Méjico, 2007.

[70] X. Wan, "Towards a Unified Approach to Simultaneous Single-Document and Multi-Document Summarizations," *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010),* pp. 1137-1145, August 2010.

[71] R. Montiel Soto, R. A. García-Hernández, Y. Ledeneva, and R. Cruz Reyes, "Comparison of three text models for automatic generation of summaries," *Procesamiento del lenguaje natural,* vol. 43, 2009.

[72] E. Greengrass. (2000). *Information Retrieval: A Survey*.

[73] S. Luke. (2013). *Essentials of Metaheuristics (second ed.)*. Available: Available for free at http://cs.gmu.edu/$\sim$sean/book/metaheuristics/

[74] T. Weise. (2009). *Global Optimization Algorithms - Theory and Application (Second ed.)*. Available: Online available at http://www.it-weise.de/projects/book.pdf

[75] S. Kirkpatrick, "Optimization by simmulated annealing," *science,* vol. 220, pp. 671-680, 1983.

[76] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*: Prentice Hall, 1995.

[77] F. Glover, *Tabu Search Fundamentals and Uses*, 1995.

[78] B. M. Batista and F. Glover, "Introducción a la Búsqueda Tabú," 2006.

[79] M. Mendoza, C. Cobos, and E. León, "Extractive Single-Document Summarization Based on Global-Best Harmony Search and a Greedy Local Optimizer," in *Advances in Artificial Intelligence and Its Applications: 14th Mexican International Conference on Artificial Intelligence, MICAI 2015, Cuernavaca, Morelos, Mexico, October 25-31, 2015, Proceedings, Part II*, O. Pichardo Lagunas, O. Herrera Alcántara, and G. Arroyo Figueroa, Eds., ed Cham: Springer International Publishing, 2015, pp. 52-66.

[80] M. Mendoza, C. Cobos, E. León, M. Lozano, F. Rodríguez, and E. Herrera-Viedma, "A New Memetic Algorithm for Multi-document Summarization Based on CHC Algorithm and Greedy Search," in *Human-Inspired Computing and Its*

Applications. vol. 8856, A. Gelbukh, F. Espinoza, and S. Galicia-Haro, Eds., ed: Springer International Publishing, 2014, pp. 125-138.

[81] C.-Y. Lin and E. Hovy, " Identifying topics by position," in *Proceedings of the Fifth conference on Applied natural language processing. San Francisco, CA, USA.*, 1997, pp. 283-290.

[82] M. A. Fattah and F. Ren, "GA, MR, FFNN, PNN and GMM based models for automatic text summarization," *Computer Speech and Language,* vol. 23, pp. 126-144, 2009.

[83] A. Bossard, M. Genereux, and T. Poibeau, "Description of the LIPN Systems at TAC 2008: Summarizing Information and Opinions," *In Notebook Papers and Results, Text Analysis Conference (TAC-2008,* 2008.

[84] J. Silla, C. Nascimento, G. L. Pappa, A. A. Freitas, and C. A. A. Kaestner, "Automatic text summarization with genetic algorithm-based attribute selection," *Lecture Notes in Artificial Intelligence,* vol. 3315, pp. 305-314, 2004

[85] R. Alguliev, R. Aliguliyev, M. Hajirahimova, and C. Mehdiyev, "MCMR: Maximum coverage and minimum redundant text summarization model," *Expert Systems with Applications,* vol. In Press, Corrected Proof, 2011.

[86] C.-Y. Lin and E. Hovy, "Identifying topics by position," *Proceedings of the fifth conference on Applied natural language processing,* pp. 283-290, 1997.

[87] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C* vol. 2: Citeseer, 1992.

[88] D. Gillick, "Sentence Boundary Detection and the Problem with the U.S," in *Proc. Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL: HLT), Companion Volume: Short Papers*, Boulder, Colorado, 2009, pp. 241-244.

[89] G. Salton, *The SMART Retrieval System---Experiments in Automatic Document Processing*: Prentice-Hall, Inc., 1971.

[90] J. Lovins, "Development of a Stemming Algorithm," *Mechanical Translation and Computational Linguistics,* vol. 11, 1968.

[91] M. F. Porter, "An algorithm for suffix stripping," *Program: Electronic Library & Information Systems,* vol. 40, pp. 211-218, 2006.

[92] H. Huang and B. Zhang, "Text Indexing and Retrieval," in *Encyclopedia of Database Systems*, ed, 2009, pp. 3055-3058.

[93] A. S. Foundation. (2011, Marzo). *Apache Lucene Core*. Available: http://lucene.apache.org/

[94] C.-Y. Lin, "Rouge: a package for automatic evaluation of summaries," in *Proceedings of the ACL-04 Workshop on Text Summarization Branches Out*, Barcelona, Spain, 2004, pp. 74-81.

[95] X. Wan, J. Yang, and J. Xiao, "Manifold-Ranking Based Topic-Focused Multi-Document Summarization."

[96] R. M. Aliguliyev, "Performance evaluation of density-based clustering methods," *Information Sciences,* vol. 179, pp. 3583-3602, 2009.