Generación automática de resúmenes extractivos genéricos de múltiples documentos basado en Word2Vec

Álvaro José Echeverría Restrepo

Anexos

Director: Ph.D. Carlos Alberto Cobos Lozada

Co-Director: Ph.D. Martha Eliana Mendoza Becerra

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información (GTI)
Área de Interés: Gestión de la Información y Sistemas Inteligentes
Popayán, Abril de 2018

Anexo A

Título	Algoritmos base: LexRank, LSA y MCMR, para la generación automática de resúmenes extractivos genéricos de múltiples documentos.
Notas	No publicado.

Algoritmos base: LexRank, LSA y MCMR, para la generación automática de resúmenes extractivos genéricos de múltiples documentos

1 ALGORITMOS BASE

Este documento es una adaptación del capítulo 3 de la tesis de grado “Algoritmo para generación automática de resúmenes extractivos genéricos de múltiples documentos basado en consensos” realizada por Fabián Anacona.

1.1 LexRank con Umbral

En [1] se plantea el Algoritmo LexRank con Umbral que se basa en el concepto de centralidad de oraciones (u oraciones sobresalientes) para así identificar las más importantes oraciones de un documento o grupo de documentos. Para esto se realiza la representación de los documentos mediante el concepto de prestigio1 en las redes sociales. Las redes sociales son representadas comúnmente mediante grafos, donde los nodos representan las entidades y los enlaces representan la relación entre cada entidad. En una red social se definen diferentes relaciones entre las entidades por ejemplo, entre personas y organizaciones; siendo estas relaciones unas más fuertes que otras.

Un documento puede representarse como una red de oraciones relacionadas; algunas comparten mayor similitud, mientras que otras pueden compartir poca información con las demás oraciones. Lo que plantearon los autores es que si una oración es muy similar a muchas oraciones, ésta se puede considerar como la más central o relevante. Para definir la centralidad de una oración hay dos cosas que se deben tener en cuenta: ¿cómo establecer la similitud entre dos oraciones? y ¿cómo calcular la centralidad global de una oración dada su similitud con otras oraciones?

Para definir la similitud entre dos oraciones, primero se representan las oraciones del conjunto de documentos en el modelo de espacio vectorial descrito en la sección 2.3.1 de la monografía. La similitud entre dos oraciones se define por el cálculo de la similitud coseno descrito en la sección 2.3.3.1 (ecuación (2-64) de la monografía); luego las oraciones son representadas como un grafo a través de una matriz de adyacencia donde cada valor corresponde a la similitud de coseno entre las oraciones de la colección de documentos (véase ecuación (2-66) de la monografía).

1Prestigio y Centralidad en esta propuesta, representan el mismo concepto con la diferencia de que el primero se define usualmente para grafos dirigidos, mientras el segundo se define para grafos no dirigidos.

La **Tabla 1** muestra un subconjunto de noticias de la colección DUC2004. Posteriormente en la **Tabla 2** se muestra la matriz de adyacencia correspondiente. Esta matriz posteriormente se usa para representar el grafo ponderado que relaciona todas las oraciones.

Tabla 1. Subconjunto de documentos del tópico d1003t de DUC 2004. Tomado de [1]

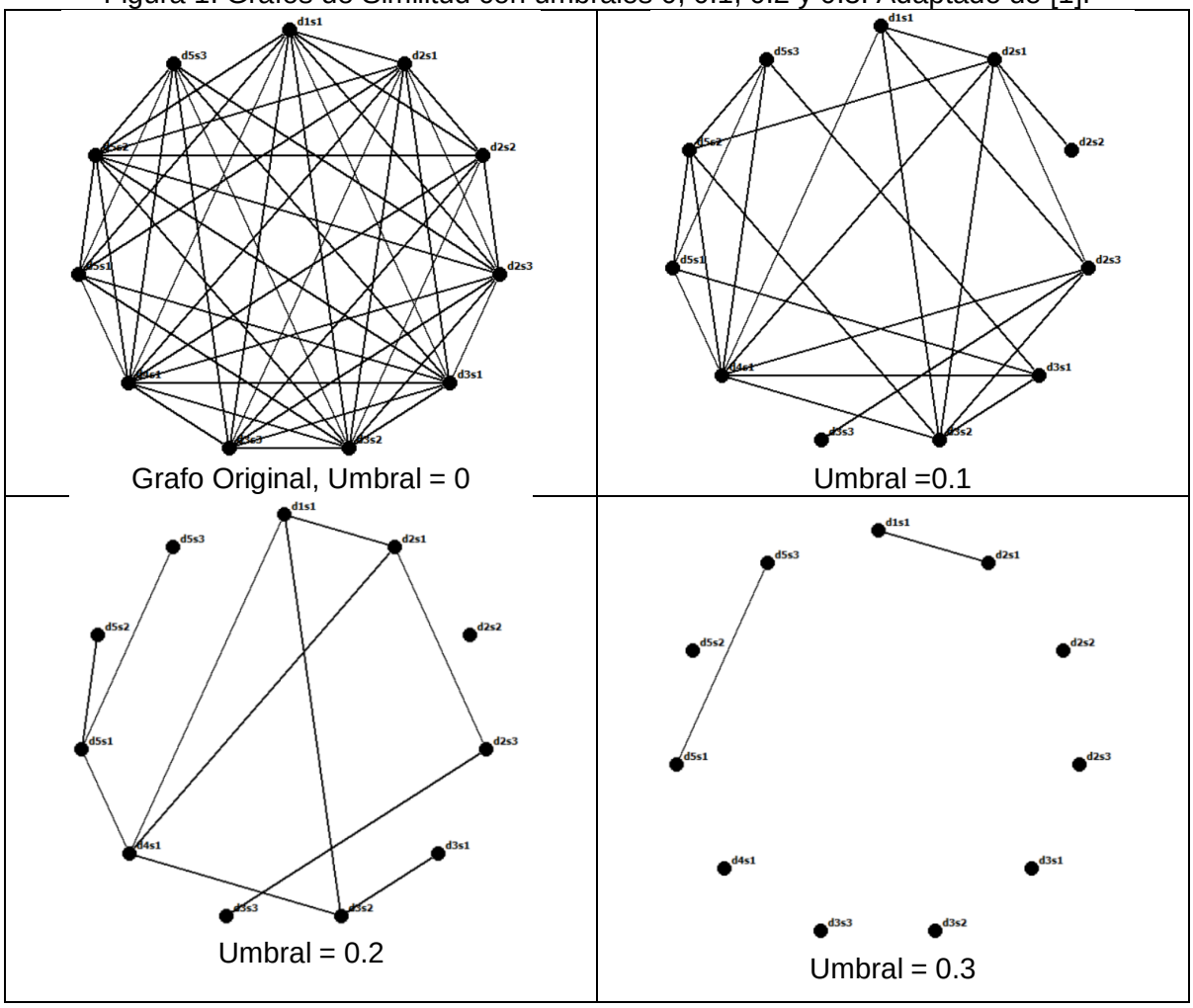
SNo	ID	Text
1	d1s1	Iraqi Vice President Taha Yassin Ramadan announced today, Sunday, that Iraq refuses to back down from its decision to stop cooperating with disarmament inspectors before its demands are met.
2	d2s1	Iraqi Vice president Taha Yassin Ramadan announced today, Thursday, that Iraq rejects cooperating with the United Nations except on the issue of lifting the blockade imposed upon it since the year 1990.
3	d2s2	Ramadan told reporters in Baghdad that “Iraq cannot deal positively with whoever represents the Security Council unless there was a clear stance on the issue of lifting the blockade off of it”
4	d2s3	Baghdad had decided late last October to completely cease cooperating with the inspectors of the United Nations Special Commission (UNSCOM), in charge of disarming Iraq’s weapons, and whose work became very limited since the fifth of August, and announced it will not resume its cooperation with the Commission even if it were subjected to a military operation.
5	d3s1	The Russian Foreign Minister, Igor Ivanov, warned today, Wednesday against using force against Iraq, which will destroy, according to him, seven years of difficult diplomatic work and will complicate the regional situation in the area.
6	d3s2	Ivanov contended that carrying out air strikes against Iraq, who refuses to cooperate with the United Nations inspectors, “will end the tremendous work achieved by the international group during the past seven years and will complicate the situation in the region.”
7	d3s3	Nevethless, Ivanov stressed that Baghdad must resume working with the Special Commission in charge of disarming the Iraqi weapons of mass destruction (UNSCOM).
8	d4s1	The Special Representative of the United Nations Secretary-General in Baghdad, Prakash Shah, announced today, Wednesday, after meeting with the Iraqi Deputy Prime Minister Tariq Aziz, that Iraq refuses to back down from its decision to cut off cooperation with the disarmament inspectors.
9	d5s1	British Prime Minister Tony Blair said today, Sunday, that the crisis between the international community and Iraq “did not end” and that Britain is still “ready, prepared, and able to strike Iraq.”
10	d5s2	In a gat hering with the press held at the Prime Minister’s office, Blair contended that the crisis with Iraq “will not end until Iraq has absolutely and unconditionally respected its commitments” towards the Unite Nations.
11	d5s3	A spokesman for Tony Blair had indicated that the British Prime Minister gave permission to British Air Force Tornado planes stationed in Kuwait to join the aerial bombardment against Iraq.

Tabla 2. Matriz de similitud de Cosenos para el subconjunto del tópico d1003t de DUC 2004. Tomado de [1]

	d1s1	d2s1	d2s2	d2s3	d3s1	d3s2	d3s3	d4s1	d5s1	d5s2	d5s3
d1s1	1	0,45	0,02	0,17	0,03	0,22	0,03	0,28	0,06	0,06	0
d2s1	0,45	1	0,16	0,27	0,03	0,19	0,03	0,21	0,03	0,15	0
d2s2	0,02	0,16	1	0,03	0	0,01	0,03	0,04	0	0,01	0
d2s3	0,17	0,27	0,03	1	0,01	0,16	0,28	0,17	0	0,09	0,01
d3s1	0,03	0,03	0	0,01	1	0,29	0,05	0,15	0,2	0,04	0,18
d3s2	0,22	0,19	0,01	0,16	0,29	1	0,05	0,29	0,04	0,2	0,03
d3s3	0,03	0,03	0,03	0,28	0,05	0,05	1	0,06	0	0	0,01
d4s1	0,28	0,21	0,04	0,17	0,15	0,29	0,06	1	0,25	0,2	0,17
d5s1	0,06	0,03	0	0	0,2	0,04	0	0,25	1	0,26	0,38
d5s2	0,06	0,15	0,01	0,09	0,04	0,2	0	0,2	0,26	1	0,12
d5s3	0	0	0	0,01	0,18	0,03	0,01	0,17	0,38	0,12	1

Este algoritmo hace uso del umbral para eliminar aquellas relaciones débiles entre las oraciones (nodos del grafo), es decir, aquellos vértices con similitud de coseno que no supera un valor establecido. La **Figura 1** muestra el efecto del umbral sobre el grafo de oraciones para diversos valores, a saber 0, 0.1, 0.2 y 0.3. Aunque en la **Tabla 3** se evidencia que existe una fuerte relación de cada nodo consigo mismo (valor de similitud de 1 en toda la diagonal), en la **Figura 1** no se muestran dichas relaciones para que el grafo sea visualmente más simple, no obstante para los cálculos respectivos se tendrán en cuenta.

Figura 1. Grafos de Similitud con umbrales 0, 0.1, 0.2 y 0.3. Adaptado de [1].



Para definir la centralidad de una oración, se cuenta el número de oraciones que conservan una similitud con esta después de haber aplicado el umbral (los vértices que no superan el umbral se establece su valor en cero para no ser tenidos en cuenta).

En la **Tabla 3** se observa cómo influye la elección del umbral en la búsqueda de la centralidad. Umbrales demasiado bajos pueden involucrar ruido en el algoritmo ya que se pueden tener en cuenta en similitudes débiles entre oraciones poco influyentes, mientras que umbrales demasiado grandes pueden eliminar muchas de las relaciones de similitud de oraciones deseadas en el resumen.

Tabla 3. Grado de Centralidad (incluye la relación del nodo consigo mismo). La oración d4s1 es la más central para los umbrales 0.1 y 0.2. Adaptado de [1].

ID	Grado (0.1)	Grado (0.2)	Grado (0.3)
d1s1	5	4	2

d2s1	7	4	2
d2s2	2	1	1
d2s3	6	3	1
d3s1	5	2	1
d3s2	7	4	1
d3s3	2	2	1
d4s1	**9**	**5**	**1**
d5s1	5	4	2
d5s2	6	2	1
d5s3	5	2	2

En éste cálculo se distingue a cada relación (vértice) como un voto para determinar el valor total de centralidad de cada oración (nodo); éste método se considera totalmente democrático debido a que cada voto tiene el mismo valor.

Sin embargo no todos las oraciones (nodos) tienen la misma importancia. Se debe determinar que oraciones tienen mayor prestigio, haciendo que el grado de centralidad de una oración tenga en cuenta además de los votos de cada nodo, de donde vienen esos votos (centralidad de los nodos que tienen relación con el nodo actual). Para esto se debe considerar que cada nodo tiene un valor de centralidad distribuido entre el nodo mismo y sus vecinos; esto se puede expresar en la siguiente ecuación:

$$p(u) = \sum_{v \,\epsilon\, adj[u]} \frac{p(v)}{\deg(v)} \qquad (1\text{-}1)$$

Donde $\boldsymbol{p(u)}$ es la centralidad del nodo $\boldsymbol{u}$, $\boldsymbol{adj[u]}$ es el conjunto de nodos que son adyacentes a $\boldsymbol{u}$, y $\boldsymbol{\deg(v)}$ es el grado del nodo $\boldsymbol{v}$. También se puede escribir esta ecuación en notación matricial de la siguiente forma:

$$p = B^T p, \text{ que es igual a: } p^T B = p^T \qquad (1\text{-}2)$$

Donde la matriz $\boldsymbol{B}$ se obtiene de la matriz de adyacencia del grafo de similitud dividiendo cada elemento por la suma de la fila correspondiente, así:

$$B(i,j) = \frac{A(i,j)}{\sum_k A(i,k)} \qquad (1\text{-}3)$$

Se debe tener en cuenta que la suma de una fila es igual al grado del nodo correspondiente. Puesto que cada oración es similar al menos a sí misma y todas las sumas de fila son distintas de cero. La ecuación (1-2) determina que $\boldsymbol{p^T}$es el vector propio izquierdo de la matriz $\boldsymbol{B}$ con el valor propio correspondiente a $\mathbf{1}$. Para garantizar que tal vector propio existe y puede ser identificado y calculado de forma única, se deben tener en cuenta los siguientes fundamentos matemáticos.

Una matriz *estocástica* $\boldsymbol{X}$, es la matriz de transición de una cadena de Markov. En esta, un elemento $\boldsymbol{X(i,j)}$ especifica la probabilidad de transición de un estado $\boldsymbol{i}$ a un estado $\boldsymbol{j}$ en la cadena de Markov correspondiente. Por los axiomas de probabilidad, todas las filas de una matriz estocástica deben sumar $\mathbf{1}$. $\boldsymbol{x^n(i,j)}$**,** da la probabilidad del estado $\boldsymbol{i}$ para

alcanzar el estado $\boldsymbol{j}$ en $\boldsymbol{n}$ transiciones. Una cadena de Markov con la matriz estocástica $\boldsymbol{X}$ converge a una distribución estacionaria sí:

$$\lim_{n\to\infty} X^n = 1^T r \qquad (1\text{-}4)$$

Donde $\mathbf{1} = (\mathbf{1}, \mathbf{1}, \dots, \mathbf{1})$, y el vector $\boldsymbol{r}$ se llama la distribución estacionaria de la cadena de Markov. Una interpretación intuitiva de la distribución estacionaria puede entenderse por el concepto de una caminata aleatoria. Cada elemento del vector $\boldsymbol{r}$ da la probabilidad asintótica de terminar en el estado correspondiente a largo plazo, independientemente del estado inicial. Una cadena de Markov es irreducible si cualquier estado es accesible desde cualquier otro estado, es decir, para todos $\boldsymbol{i, j}$ existe un $\boldsymbol{n}$ de tal que $\boldsymbol{X^n}\ (\boldsymbol{i, j}) \neq \mathbf{0}$. Una cadena de Markov es aperiódica si para todo $\boldsymbol{i}$, $\mathbf{gcd}\{\boldsymbol{n} : \boldsymbol{X^n(i, i)} > \mathbf{0}\} = \mathbf{1}$. Por el teorema de Perron-Frobenius, una cadena de Markov irreducible y aperiódica converge a una distribución estacionaria única.

Si una cadena de Markov tiene componentes reducibles o periódicos, un caminante aleatorio puede atascarse en estos componentes y nunca visitar las otras partes del grafo.

Dado que la matriz de similitud $\boldsymbol{B}$ en la ecuación (1-2) satisface las propiedades de una matriz estocástica, se puede tratar como una cadena de Markov. El vector de centralidad $\boldsymbol{P}$ corresponde a la distribución estacionaria de $\boldsymbol{B}$. Sin embargo, se debe asegurar que la matriz de similitud sea siempre irreductible y aperiódica. Para resolver este problema, se reserva una baja probabilidad para saltar a cualquier nodo en el grafo. De esta manera el caminante puede "escapar" de componentes periódicos o desconectados, lo que hace que el grafo sea irreducible y aperiódico. Si se asigna una probabilidad uniforme para saltar a cualquier nodo en el grafo, se obtiene la siguiente versión modificada de la ecuación (1-1)(1-1), que se conoce como el algoritmo PageRank.

$$p(u) = \frac{d}{N} + (1-d)\sum_{v\ \epsilon\ adj[u]} \frac{p(v)}{\deg(v)} \qquad (1\text{-}5)$$

Donde $\boldsymbol{N}$ es el número total de nodos en el grafo y $\boldsymbol{d}$ es un "factor de amortiguamiento", que típicamente se elige en el intervalo [0.1, 0.2]. La ecuación (1-5) se puede escribir en forma matricial de la siguiente manera:

$$p = [dU + (1-d)B]^T\, p \qquad (1\text{-}6)$$

Donde $\boldsymbol{U}$ es una matriz cuadrada de NxN con todos los elementos iguales a $\frac{\mathbf{1}}{\boldsymbol{N}}$ **.** El kernel de transición $[\boldsymbol{dU} + (\mathbf{1} - \boldsymbol{d})\boldsymbol{B}]$ de la cadena de Markov resultante es una mezcla de dos núcleos (kernels) $\boldsymbol{U}$ y $\boldsymbol{B}$. Una caminata (recorrido) aleatoria en esta cadena de Markov elige uno de los estados adyacentes del estado actual con probabilidad $\mathbf{1} - \boldsymbol{d}$, o salta a cualquier estado en el grafo, incluyendo el estado actual, con probabilidad $\boldsymbol{d}$. La fórmula PageRank fue propuesto por primera vez para el cálculo del prestigio de una página y aún hoy es usada por Google.

La propiedad de convergencia de las cadenas de Markov también nos proporciona un algoritmo iterativo simple, llamado el método de potencia (o PowerMethod), para calcular la distribución estacionaria (ver **Algoritmo 1**). El algoritmo comienza con una distribución uniforme. En cada iteración, el vector propio se actualiza multiplicándolo con la transpuesta de la matriz estocástica. Dado que la cadena de Markov es irreducible y aperiódica, se garantiza que el algoritmo termina.

Algoritmo 1. Método de Potencia para el cálculo de la distribución estacionaria de una Cadena de Markov, Adaptado de [1].

Entrada: Una matriz M estocástica, irreducible y aperiódica
Entrada: Tolerancia de error £
Salida: Vector propio p
01 $p_o = \frac{1}{N} 1$
02 $t = 0$
03 repita
04 $t = t + 1$
05 $p_t = M^T P_{t-1}$
06 δ = $||p_t - p_{t-1}||$
07 hasta que δ < £;
08 retorne p_t

A diferencia del método original de PageRank, el grafo de similitud para oraciones no es dirigido, ya que la matriz de similitud de cosenos es simétrica. Sin embargo, esto no hace ninguna diferencia en el cálculo de la distribución estacionaria. A continuación se muestra el **Algoritmo 2** para el cálculo de las puntuaciones LexRank para un determinado conjunto de oraciones.

Algoritmo 2. Cálculo de puntuación de LexRank, Adaptado de [1].

```
Entrada: Arreglo S de n oraciones, umbral t, valor del factor de amortiguamiento (dampingFactor)
Salida: Arreglo L con los scores definidos por LexRank para cada oración
Arreglo CosineMatrix[n][n];
Arreglo L[n];
01 Para i=1 hasta n haga
02     suma=0
03     Para j=1 hasta n haga
04         CosineMatrix[i][j] = idf-modified-cosine(S[i],S[j]);
05         Si CosineMatriz[i][j] > t haga
06             CosineMatrix[i][j] = 1;
07             suma++;
08         Si No
09             CosineMatriz[i][j] = 0;
10         Fin Si
```

```
11        Fin Para
15 Fin Para
16 Para i=1 hasta n haga
17        Para j=1 hasta n haga
18                CosineMatriz[i][j] = CosineMatriz[i][j] / suma;
19        Fin Para
20 Fin Para
21 Para i=1 hasta n haga
22        Para j=1 hasta n haga
23                CosineMatriz[i][j] = (dampingFactor/n) + (1- dampingFactor)*
CosineMatriz[i][j];
24        Fin Para
25 Fin Para
26 L = PowerMethod(CosineMatrix, n, £ ); //Algoritmo 1 previamente presentado
27 retorne L;
```

A continuación se presenta una descripción detallada de la ejecución del algoritmo LexRank a partir de la matriz de similitudes de la **Tabla 2**. La primera gran tarea que se hace es aplicar el umbral (líneas 1 a 15 del **Algoritmo 2**). El resultado de esta tarea usando un umbral de 0.1 se presenta en la **Tabla 4**.

Tabla 4. Muestra la similitud entre las oraciones que superan el umbral de 0.1.

	d1s1	d2s1	d2s2	d2s3	d3s1	d3s2	d3s3	d4s1	d5s1	d5s2	d5s3	Grado
d1s1	1	1	0	1	0	1	0	1	0	0	0	5
d2s1	1	1	1	1	0	1	0	1	0	1	0	7
d2s2	0	1	1	0	0	0	0	0	0	0	0	2
d2s3	1	1	0	1	0	1	1	1	0	0	0	6
d3s1	0	0	0	0	1	1	0	1	1	0	1	5
d3s2	1	1	0	1	1	1	0	1	0	1	0	7
d3s3	0	0	0	1	0	0	1	0	0	0	0	2
d4s1	1	1	0	1	1	1	0	1	1	1	1	9
d5s1	0	0	0	0	1	0	0	1	1	1	1	5
d5s2	0	1	0	0	0	1	0	1	1	1	1	6
d5s3	0	0	0	0	1	0	0	1	1	1	1	5

Después se debe normalizar la matriz, dividiendo cada elemento de esta por el grado de la fila en la que se encuentra (líneas 16 al 20 del Algoritmo 2) para que cada fila pueda verse como un vector de cambios de estado con probabilidades que sumadas dan 1. Como resultado se obtienen los valores mostrados en la **Tabla 5.** En este punto la matriz se interpreta como una matriz de cambios de estado, por ejemplo para cambiar del estado d1s1 al estado d2s1 existe una probabilidad de 0.2 y para cambiar al estado d2s3 también existe una probabilidad de 0.2. Como se puede observar, la suma de las probabilidades de los cambios de estado en todas las filas suma 1.

Tabla 5. Matriz de Similitudes como Matriz de cambios de estado (matriz estocástica).

	d1s1	d2s1	d2s2	d2s3	d3s1	d3s2	d3s3	d4s1	d5s1	d5s2	d5s3
d1s1	0,2	0,2	0	0,2	0	0,2	0	0,2	0	0	0
d2s1	0,143	0,143	0,143	0,143	0	0,143	0	0,143	0	0,143	0
d2s2	0	0,5	0,5	0	0	0	0	0	0	0	0
d2s3	0,167	0,167	0	0,167	0	0,167	0,167	0,167	0	0	0
d3s1	0	0	0	0	0,2	0,2	0	0,2	0,2	0	0,2

d3s2	0,143	0,143	0	0,143	0,143	0,143	0	0,143	0	0,143	0
d3s3	0	0	0	0,5	0	0	0,5	0	0	0	0
d4s1	0,111	0,111	0	0,111	0,111	0,111	0	0,111	0,111	0,111	0,111
d5s1	0	0	0	0	0,2	0	0	0,2	0,2	0,2	0,2
d5s2	0	0,167	0	0	0	0,167	0	0,167	0,167	0,167	0,167
d5s3	0	0	0	0	0,2	0	0	0,2	0,2	0,2	0,2

El siguiente paso es garantizar que la matriz estocástica sea irreducible y aperiódica. Esto se logra aplicando un factor de amortiguamiento a cada valor de la matriz (líneas 21 a 25 del Algoritmo 2). Usando para este ejemplo un factor de amortiguamiento de 0.15, la celda d1s1 - d2s1 es igual a $a_{12} = 0.15 * \frac{1}{11} + (1 - 0.15)0.2 = 0{,}1836$. La Matriz resultante se observa en la **Tabla 6**.

Tabla 6. Matriz estocástica, irreducible y aperiódica sobre la cual ya se pueden calcular los valores de prestigio o centralidad de cada oración.

	d1s1	d2s1	d2s2	d2s3	d3s1	d3s2	d3s3	d4s1	d5s1	d5s2	d5s3
d1s1	0,183	0,183	0,013	0,183	0,013	0,183	0,013	0,183	0,013	0,013	0,013
d2s1	0,135	0,135	0,135	0,135	0,013	0,135	0,013	0,135	0,013	0,135	0,013
d2s2	0,013	0,438	0,438	0,013	0,013	0,013	0,013	0,013	0,013	0,013	0,013
d2s3	0,155	0,155	0,013	0,155	0,013	0,155	0,155	0,155	0,013	0,013	0,013
d3s1	0,013	0,013	0,013	0,013	0,183	0,183	0,013	0,183	0,183	0,013	0,183
d3s2	0,135	0,135	0,013	0,135	0,135	0,135	0,013	0,135	0,013	0,135	0,013
d3s3	0,013	0,013	0,013	0,438	0,013	0,013	0,438	0,013	0,013	0,013	0,013
d4s1	0,108	0,108	0,013	0,108	0,108	0,108	0,013	0,108	0,108	0,108	0,108
d5s1	0,013	0,013	0,013	0,013	0,183	0,013	0,013	0,183	0,183	0,183	0,183
d5s2	0,013	0,155	0,013	0,013	0,013	0,155	0,013	0,155	0,155	0,155	0,155
d5s3	0,013	0,013	0,013	0,013	0,183	0,013	0,013	0,183	0,183	0,183	0,183

Luego se hace el llamado al método PowerMethod (Algoritmo 1) usando una tolerancia de error específico de 0.0001. En la **Tabla 7** se evidencia como el vector es inicializado de manera uniforme, es decir, cada oración tienen la misma centralidad, en este ejemplo es 1/11 = 0.091. Después el método va iterando hasta que en la iteración 7 encuentra una solución estable donde la oración más central es d4s1 con un valor de 0.139.

Tabla 7. Resultado de iteraciones de PowerMethod en el vector de centralidad para cada una de las oraciones.

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
d1s1	0,091	0,073	0,082	0,083	0,084	0,084	0,084	0,084
d2s1	0,091	0,124	0,121	0,12	0,12	0,119	0,119	0,118
d2s2	0,091	0,063	0,056	0,052	0,05	0,05	0,049	0,049
d2s3	0,091	0,111	0,109	0,108	0,107	0,106	0,106	0,105
d3s1	0,091	0,08	0,08	0,08	0,081	0,081	0,082	0,082
d3s2	0,091	0,101	0,108	0,11	0,111	0,111	0,111	0,111
d3s3	0,091	0,065	0,057	0,053	0,052	0,051	0,05	0,05
d4s1	0,091	0,132	0,135	0,137	0,138	0,139	0,139	0,139
d5s1	0,091	0,081	0,08	0,08	0,081	0,081	0,082	0,082
d5s2	0,091	0,088	0,094	0,095	0,095	0,096	0,096	0,096
d5s3	0,091	0,081	0,08	0,08	0,081	0,081	0,082	0,082

Finalmente las oraciones se ordenan de mayor a menor dependiendo de la centralidad correspondiente y se incluyen en el resumen en el siguiente orden: d4s1 (0.139), d2s1 (0.118), d3s2 (0.111), d2s3 (0.105), d5s2 (0.096), d1s1 (0.084), d5s1 (0.0824), d5s3 (0.0824), d3s1 (0.0823), d3s3 (0.05) y d2s2 (0.049) hasta que se complete el tamaño máximo del resumen.

1.2 Análisis Semántico Latente (LSA o LSI)

El algoritmo de Análisis semántico latente (Latent Semantic Analysis, LSA) introducido en [2] y basado en la propuesta de [3] es una técnica matemática para extraer e inferir relaciones contextuales entre un conjunto de palabras [4].

El LSA está basado en la descomposición de valores singulares (Singular Value Decomposition, SVD) descrito en [5] y su procedimiento se describe a continuación:

1) Primero se representan las oraciones del conjunto de documentos mediante el modelo espacio vectorial descrito en la sección 2.3.1 y para determinar el peso de los términos en las oraciones se usa el esquema tf-isf de la ecuación (2-63).

2) Se obtiene la matriz $\boldsymbol{A}$ de $\boldsymbol{m} \times \boldsymbol{n}$ ($\boldsymbol{m}$ términos y $\boldsymbol{n}$ oraciones) donde $\boldsymbol{m} \geq \boldsymbol{n}$ ecuación (2-65). Esta matriz es la igual a la matriz del paso anterior si $\boldsymbol{m}$ es mayor que $\boldsymbol{n}$, de lo contrario la matriz $\boldsymbol{A}$ se transpone.

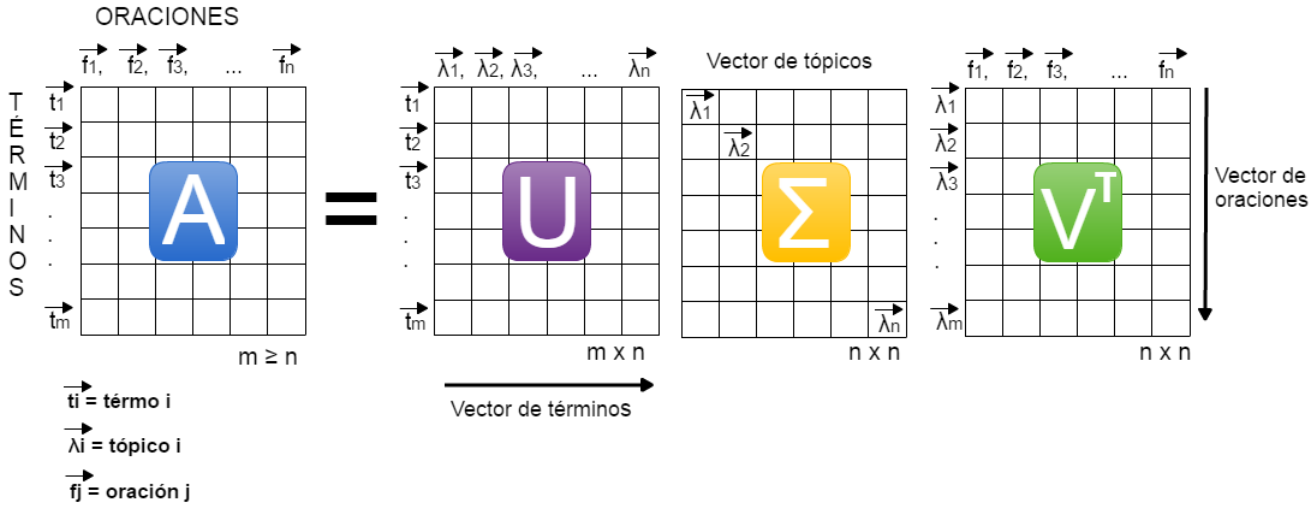
3) Luego se aplica la descomposición en valores singulares (SVD) a la matriz $\boldsymbol{A}$, como lo ilustra la ecuación que se muestra a continuación.

$$A_{m\times n} = U_{m\times r}\, \Sigma_{r\times r}\, \mathrm{V}^{T}_{r\times n} \qquad (2\text{-}1)$$

Dónde $U = [u_{ij}]$ es una matriz de columnas ortonormales de $m \times r$ cuyas columnas son llamadas vectores singulares de izquierda, $\Sigma = \mathrm{diag}\,(\sigma_1, \sigma_2, \ldots, \sigma_r)$ es una matriz diagonal de $r \times r$ donde r es el menor entre m y n, cuyos elementos diagonales son valores singulares no negativos (representan los tópicos) en orden descendente ($\sigma_1 \geq \sigma_2, \ldots, \sigma_k \geq \sigma_{k+1}, \ldots, = \sigma_{r-1} = \sigma_r = 0$) y $V = [v_{ij}]$ es una matriz ortonormal de $r \times n$, cuyas columnas se denominan vectores singulares derechos.

Como se explica en [3], SVD produce un mapeo del espacio m-dimensional generado por los vectores tf-isf de la matriz A al espacio vectorial singular r-dimensional derivado de la estructura semántica latente del conjunto de oraciones. El mapeo proyecta cada vector columna j (oración) de la matriz A al vector columna latente $V_j = \left[v_{1j}, v_{2j}, \ldots, v_{rj}\right]^T$ de la Matriz V^T. También se mapea cada vector fila i (término) de la matriz A al vector fila latente $U_i = \left[u_{i1}, u_{i2,\ldots,} u_{ir}\right]$ de la Matriz U, Como se muestra en la **Figura 2**.

Figura 2 Descomposición en valores singulares (SVD). Adaptado de [6].



Al aplicar SVD se logra detectar interrelaciones entre los términos y agruparlos semánticamente en tópicos (vectores $\vec{\lambda}$ de la matriz Σ) para posteriormente representar los términos y las oraciones en función de esos tópicos en las matrices U y V^T. Adicionalmente SVD ordena los vectores de tópicos por su importancia, siendo el primero ($\vec{\lambda}_1$) el de mayor importancia. Así la tarea de encontrar las mejores frases para incluirlas en el resumen se resume en seleccionar las oraciones más similares a los vectores de tópicos (similitud coseno, véase la sección 2.3.3.1 de la monografía) sin repetirlas hasta completar el tamaño del resumen.

Teniendo en cuenta lo anterior, se presenta el Algoritmo 3 de LSA aplicando SVD en generación automática de resúmenes de múltiples documentos.

Algoritmo 3. LSA basado en SVD para la generación de resúmenes de múltiples documentos. Adaptado de [2]

Paso 1: Descomponer el conjunto de documentos D en oraciones individuales en el espacio vectorial descrito en la sección 2.3.1 de la monografía, y usar las oraciones para formar el conjunto S de oraciones candidatas (oraciones que pueden integrar el resumen).

Paso 2: Construir la Matriz A de términos por oraciones según la ecuación (2-65) de la monografía.

Paso 3: Aplicar SVD a la Matriz A para obtener las frases latentes de la Matriz V^T.

Paso 4: Inicializar $k = 1$.

Paso 5: Seleccionar la oración que tiene el valor más alto para el $k - ésimo$ tópico, es decir la oración j más similar al tópico $\vec{\lambda}_k$. Si la oración ya existe en la selección escoger la siguiente oración que más aporte al tópico $\vec{\lambda}_k$.

Paso 6: Si k alcanza el número predefinido (o se completó el tamaño del resumen) terminar la operación, sino incrementar k en uno y volver al paso 5.

Paso 7: Aplicar la similitud de coseno (ecuación (2-64) de la monografía) entre las oraciones elegidas anteriormente y el vector centroide (ecuación (2-67) de la monografía).

Paso 8: Ordenar las oraciones en el resumen de mayor a menor similitud al vector centroide.

1.3 Máxima Cobertura y Mínima Redundancia (MCMR)

El algoritmo Máxima Cobertura Mínima Redundancia (MCMR) es presentado es 2011 [7] partiendo de la premisa que en la generación automática de resúmenes de múltiples documentos se busca maximizar la similitud del resumen con la colección de documentos. Representando tanto el resumen como la colección; como vectores en el modelo espacio vectorial (véase sección 2.3.1 de la monografía), bajo la restricción de la longitud máxima del resumen que se desea obtener (ver ecuación (3-1)).

(3-1)

$$\text{Maximizar } sim\left(\vec{D}, \vec{S}\right)$$
$$sujeto\ a\ len(S) \leq L$$

Donde $len(S)$ es la longitud del resumen $S\ \epsilon\ D$, y L es el tamaño máximo del resumen.

Debido a la generalidad de las ecuaciones, los autores redefinieron la función en base a cumplir con dos objetivos, maximizar la cobertura y minimizar la redundancia. Siendo la cobertura, la similitud de las oraciones del resumen conjunto de documentos, y la redundancia, la similitud entre las oraciones del resumen. Dicha transformación se expresa formalmente en la siguiente ecuación (3-2).

(3-2)

$$\text{Maximizar } f = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left[sim\left(\vec{D}, \vec{s}_i\right) + sim\left(\vec{D}, \vec{s}_j\right) - sim(\vec{s}_i, \vec{s}_j)\right] x_{ij}$$

$$\text{sujeto a } \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left[len\left(s_i\right) + len\left(s_j\right)\right] x_{ij} \leq L$$

$$y\ x_{ij} \epsilon \{0,1\},\ \forall_{i,j}$$

Donde $\vec{D}$ es el vector que representa a la colección de documentos en el modelo espacio vectorial ecuación (2-67) de la monografía, n corresponde al número de oraciones en la colección de documentos, $\vec{s}_i$ y $\vec{s}_j$ son vectores que representan a la oración i y j en el modelo espacio vectorial (véase sección 2.3.1 de la monografía), la $sim\left(\vec{D},\vec{s}_i\right)$ y $sim\left(\vec{D},\vec{s}_j\right)$ es la similitud de cosenos (véase sección 2.3.3.1) entre el vector de la colección de documentos y el vector de la oración i y la similitud de cosenos entre el vector de la colección de documentos y el vector de la oración j respectivamente. Estos dos componentes se consideran la cobertura. La $sim(\vec{s}_i,\vec{s}_j)$ es la similitud de cosenos entre las oraciones del resumen que se conoce como redundancia (lo que se desea minimizar). Y X_{ij} es una variable binaria que toma el valor de 1 si las dos oraciones s_i y s_j son incluidas en el resumen, de lo contrario es 0 haciendo que solo sean tenidas en cuenta aquellas oraciones que han sido incluidas en el resumen.

El objetivo [7] es encontrar aquella combinación de X_{ij} que represente la mayor cobertura y la menor redundancia sin superar el tamaño máximo del resumen L. La función objetivo descrita en la ecuación (3-2) es un problema clásico de optimización lineal binaria que garantiza que el contenido más relevante de la colección de documentos estará incluido resumen, esto lo garantiza el primer y segundo término de la ecuación. Y el último término asegura que el resumen contenga la variedad presente en el conjunto de documentos y así no involucrar la misma información.

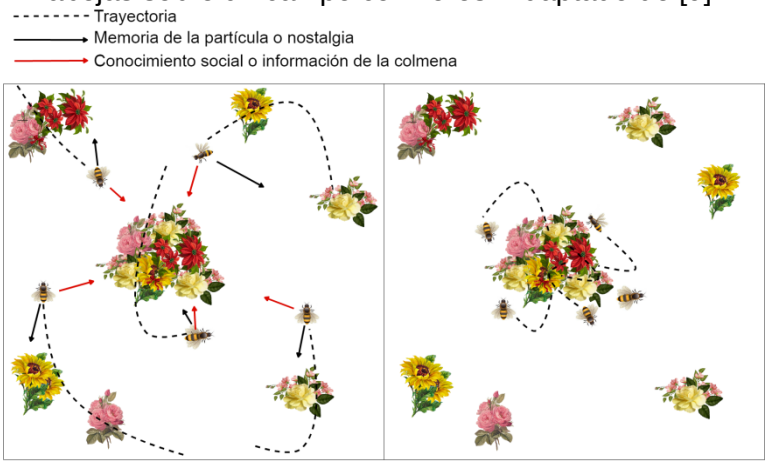
Para hacer frente a este problema, existen varios algoritmos para la optimización lineal binaria. Pero los autores prefieren hacer uso de uno en particular; el algoritmo de Optimización por Enjambre de Partículas (Particle Swarm Optimization, PSO) propuesto originalmente por Kennedy y Eberhart [8], es un algoritmo que trata de imitar el comportamiento de algunos grupos de animales con la capacidad de dispersarse e integrarse para optimizar la búsqueda de su alimento.

En PSO el término *partícula* o *agente* representa a los peces, pájaros, abejas, hormigas o cualquier otro grupo de individuos que presenten un comportamiento social semejante. Cada partícula se mueve hacia un objetivo común en dos dimensiones bajo la información individual llamada memoria autobiográfica de la partícula o nostalgia; y la información del enjambre. Ésta filosofía es tomada para crear el método de optimización y se extiende a un espacio N-dimensional de acuerdo al problema abordado. La posición instantánea de cada partícula de la población en el espacio de búsqueda representa una posible solución. El proceso consiste en mover cada partícula dentro del espacio de soluciones variando la velocidad de acuerdo a su velocidad actual, a la memoria de la partícula (nostalgia) y a la información del enjambre; utilizando una función de fitness para determinar la calidad de la posición de cada partícula.

Para ilustrar este método la **Figura 3** muestra el movimiento de un enjambre de abejas sobre un campo con flores. “(a) En su desplazamiento, las abejas son atraídas hacia las zonas de mayor concentración de flores encontradas personalmente por cada individuo (memoria) y por el conjunto del enjambre (cooperación). (b) Una vez que las abejas han

sido atraídas a la zona con mayor concentración de flores, que equivale en términos de PSO a la convergencia hacia una solución global, éstas permanecen sobrevolando dicha zona con velocidades muy reducidas" [9].

Figura 3 Modelado del PSO utilizando como símil el movimiento de un enjambre de abejas sobre un campo con flores. Adaptado de [9].



El algoritmo PSO para problemas de optimización global usa un enjambre de N_{sw} partículas, donde cada partícula i del enjambre está asociada a una posición en el espacio de búsqueda (N-dimensional) $\mathrm{p}_i(\mathrm{t}) = [\mathrm{p}_{i1}(\mathrm{t}),\ldots,\mathrm{p}_{in}(\mathrm{t})],\mathrm{i} = 1,\ldots,\mathrm{N}_{sw}$. Para cada partícula la mejor posición recorrida y la velocidad son representadas respectivamente por $\mathrm{p}_i^{best} = [\mathrm{p}_{i1}^{best}, \ \ldots \ ,\mathrm{p}_{in}^{best}]$ y $v_i(t) = [v_{i1}(t), \ \ldots \ , v_{in}(t)]$. La posición correspondiente al mejor valor encontrado por el enjambre es $g^{best} = [\mathrm{f}_{\boldsymbol{\alpha}}(\mathrm{p}_1(\boldsymbol{t})), \ \ldots \ ,\mathrm{f}_{\boldsymbol{\alpha}}(\mathrm{p}_{Nsw}(\boldsymbol{t}))]$. Donde $\mathrm{f}_{\boldsymbol{\alpha}}(\cdot)$ es la función objetivo.

Para iniciar, el algoritmo asigna valores aleatorios a las partículas para su posición y su velocidad teniendo en cuenta que hay posiciones y velocidades máximas y mínimas permitidas. Esto está descrito en las ecuaciones (3-3) y (3-4).

(3-3)

$$p_{ij}(0) = \ p_{min} + (p_{max} - p_{min}) \cdot r_1$$

(3-4)

$$v_{ij}(0) = \ v_{min} + (v_{max} - v_{min}) \cdot r_2$$

Las ecuaciones (3-5) y (3-6) son usadas para actualizar los vectores de velocidad y posición de las partículas durante el proceso iterativo.

(3-5)

$$v_{ij}(t+1) = w \cdot \ v_{ij}(t) + c_1 \cdot r_1 \cdot \left(p_{ij}^{best}(t) - p_{ij}(t)\right) + c_2 \cdot r_2 \cdot \left(g_j^{best}(t) - p_{ij}(t)\right)$$

(3-6)

$$\mathrm{p}_{ij}(t+1) = \mathrm{p}_{ij}(t) + v_{ij}(t+1)$$

Donde $\mathrm{p}_{ij}(t)$ y $v_{ij}(t)$ representan la posición y la velocidad de la i-ésima partícula en la j-ésima dimensión (para $j = 1,2, \ldots, n$) en la iteración t ($t = 0,1, \ldots, Numero\ Iteraciones$). Los valores r_1 y r_2 son números aleatorios uniformemente distribuidos en el intervalo $[0,1]$, c_1 es el factor de aceleración cognitiva que define el movimiento de la partícula bajo la influencia de su propia memoria; c_2 es el factor de aceleración social y determina el grado en el que influyen sobre el movimiento de la partícula la información de los individuos del enjambre. $\mathrm{p}_{ij}^{best}(t)$ es el vector con la mejor posición encontrada por la partícula i en el proceso iterativo en la dimensión j y $g_j^{best}(t)$ es el vector con la mejor posición encontrada por el enjambre durante el proceso iterativo de búsqueda en la dimensión j.

Un punto delicado del algoritmo es establecer los valores adecuado para v_{max} y v_{min} debido a que si el valor de v_{max} es demasiado grande, las partículas pueden sobrepasar e ignorar continuamente la zona con la mejor solución (solución global); si el valor de v_{min} toma valores muy pequeños, las partículas explorarán un espacio de soluciones muy pequeño quedando atrapadas en soluciones locales (máximo local de la función de fitness).

El peso inercial w se calcula para controlar la influencia de la velocidad anterior respecto a la velocidad actual, y así mejorar la convergencia de la partícula. Con este factor se regula la relación entre exploración del espacio de búsqueda y explotación de las soluciones locales o global, dado por el segundo y tercer sumando descrito en la ecuación (3-5).

A continuación se muestran las ecuaciones (3-7) (3-8) para el cálculo del peso de inercia w. Teniendo en cuenta que una buena convergencia se puede alcanzar al hacer la aceleración y las constantes de inercia dependientes. Su relación se muestra en la siguiente ecuación con un parámetro intermedio φ.

(3-7)

$$w = \frac{1}{\varphi - 1 + \sqrt{\varphi^2 - 2\varphi}}$$

(3-8)

$$c_1 = c_2 = \varphi w$$

La principal diferencia entre el PSO binario con la versión continua es que las velocidades de las partículas están definidas mediante la probabilidad de que un bit cambie a uno. Por lo tanto, la velocidad debe estar restringida en el intervalo [0, 1]. En el PSO binario, la ecuación (3-5) para actualizar la velocidad permanece igual, pero la ecuación (3-6) para actualizar la posición se redefine en la ecuación (3-9).

(3-9)

$$p_{ij}(t+1) = \begin{cases} 1 & if\ rand_j < sigm\left(v_{ij}(t+1)\right) \\ 0 & de\ otro\ modo \end{cases}$$

Donde $rand_j$ es un escalar aleatorio seleccionado en un intervalo uniforme [0,1] y $sigm(\cdot)$ es la función sigmoidea para transformar la velocidad a un intervalo de probabilidad [0,1] como se define en la ecuación (3-10).

(3-10)

$$sigm\left(v_{ij}(t+1)\right) = \frac{1}{1 + e^{\left(-v_{ij}(t+1)\right)}}$$

A continuación se presenta el **Algoritmo 3** que describe los pasos del algoritmo PSO binario usado en la propuesta de Aliguliyev et al. [7].

Algoritmo 3. Descripción de PSO binario. Adaptado de [7].

Paso 1	Inicializar: los parámetros y población con posiciones aleatorias y velocidades usando ecuaciones (3-3) y (3-4).
Paso 2	Evaluar: Evaluar el valor del fitness (función objetivo) para cada partícula.
Paso 3	Encontrar el mejor valor individual: si el valor actual del fitness de la partícula i es mejor que su mejor valor de fitness (mejor marca personal), establezca el valor actual como nuevo mejor fitness de la partícula i, de lo contrario no haga nada.
Paso 4	Encontrar el mejor valor global: si algún valor de una partícula se actualiza y es mejor que el valor global actual, coloque como nuevo mejor valor global el valor actual (fitness) de la partícula i.
Paso 5	Actualizar velocidad y posición: Actualice la velocidad y muévase a la siguiente posición según las ecuaciones (3-5) y (3-9).
Paso 6	Revisar el criterio de parada: si el número máximo de iteraciones es alcanzado entonces parar, en otro caso vuelva al paso 2 y repita el proceso.

2 Bibliografía

[1] G. Erkan and D. R. Radev, "LexRank: graph-based lexical centrality as salience in text summarization," *J. Artif. Int. Res.,* vol. 22, pp. 457-479, 2004.

[2] D. Ai, Y. Zheng, and D. Zhang, "Automatic text summarization based on latent semantic indexing," *Artificial Life and Robotics,* vol. 15, pp. 25-29, 2010// 2010.

[3] Y. Gong and X. Liu, "Generic text summarization using relevance measure and latent semantic analysis," presented at the Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, New Orleans, Louisiana, USA, 2001.

[4] M. Mendoza and E. Leon, "Una revisión de la generación automática de resúmenes extractivos," *UIS Ingenierías,* vol. 12, pp. 7-27, 2015.

[5] R. P. Mera and H. J. Martínez, "Introducción al Álgebra Lineal Numérica," *Universidad del Cauca,* vol. 4.3, pp. 1-137, 1998.

[6] J. Steinberger, K. Je, #382, and ek, "Text summarization and singular value decomposition," presented at the Proceedings of the Third international conference on Advances in Information Systems, Izmir, Turkey, 2004.

[7] R. M. Alguliev, R. M. Aliguliyev, M. S. Hajirahimova, and C. A. Mehdiyev, "MCMR: Maximum coverage and minimum redundant text summarization model," *Expert Systems with Applications,* vol. 38, pp. 14514-14522, 2011.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 1995, pp. 1942-1948 vol.4.

[9] J. R. P. López, "Contribución a los métodos de optimización basados en procesos naturales y su aplicación a la medida de antenas en campo próximo," *Departamento de Ingeniería de Comunicaciones, Universidad de Cantabria, Santander,* 2005.

Anexo B

Título	Artículo con los resultados del algoritmo de generación automática de resúmenes extractivos genéricos de múltiples documentos basado en Word2Vec.
Notas	No publicado.

Anexo C

Título Manual de instalación para para la configuración del entorno de desarrollo de Herramienta que implementa los algoritmos con el esquema de representación basado en Word2Vec.

Notas No publicado.