DESARROLLO DE ALGORITMOS PARA EL DESCUBRIMIENTO DE PATRONES SECUENCIALES MAXIMALES

René Arnulfo García Hernández

Tesis sometida como requisito parcial para obtener el grado de Doctor en Ciencias en la especialidad de Ciencias Computacionales en el Instituto Nacional de Astrofísica, Óptica y Electrónica, Septiembre 2007.

Asesores:
Dr. José Francisco Martínez Trinidad
Dr. Jesús Ariel Carrasco Ochoa

Dedico este trabajo a:

A mi madre, Guadalupe Hernández García
A mi padre, Leopoldo René García Castro
A mi hermana, Susana García Hernández

Detrás de una gran familia, siempre hay una gran madre.

A Yulia Ledeneva, por su amor y compromiso

A Dios, porque esto es una muestra de su existencia

Agradecimientos

Confieso que me entusiasma haber llegado a este punto de la investigación en donde me es posible expresar mi agradecimiento a quienes provocaron, fomentaron y concretaron este sueño que tomó días, meses y años.

Mi más grande agradecimiento es para mis asesores, el Dr. José Francisco Martínez Trinidad y el Dr. Jesús Ariel Carrasco Ochoa, por asumir el gran reto de mi formación profesional, mediante este agradecimiento les quiero reconocer el tiempo, paciencia y recursos que me han confiado. También les agradezco a mis asesores por ser para mí un ejemplo a seguir en la ética profesional de un investigador, ya que en cada una de las etapas de la investigación creyeron en mis ***eficientes*** ideas. ¡GRACIAS!.

También quiero agradecer a un investigador que he admirado desde años atrás, y a quien agradezco sus comentarios propositivos en mi tesis, al Dr. Alexander Gelbuckh.

Además quiero agradecer al Dr. Manuel Montes y Gómez quien me dio una gran cantidad de bibliografía, incluso fue de ahí donde encontré el artículo de Ahonen, el cual formó la base inicial de este trabajo. También estoy agradecido por sus valiosos comentarios propositivos para este trabajo. En este sentido, también quiero agradecer al Dr. Luís Villaseñor Pineda, Dr. Jesús González Bernal y al Dr. Aurelio López López por la revisión de la tesis, así como por los atinados comentarios sobre el trabajo.

En particular agradezco a un hombre leal y sencillo que me brindo su amistad y tiempo, Luís Alberto Morales. También les quiero dar las gracias a mis amigos de caminatas nocturnas: Luís Alberto Morales, Iván Candelas y Christopher. También quiero agradecer a mis compañeros del INAOE tanto de maestría como de doctorado por haberme apoyado con sus comentarios.

Todo esto no sería posible sin la ayuda de la beca económica otorgada por CONACYT con quienes estaré agradecido por siempre. De igual forma, estaré en deuda con mi institución

INAOE por haberme formado con programas de estudio de calidad y por su excelente trato personal hacia los alumnos.

Abstract

Information of a document is described by words in a sequential way. From this point of view, the knowledge transmitted in a text is sequentially structured by its author. Thus, text it is not only useful for expressing the author's ideas, but also it is useful for discovering new knowledge from the frequent sequential order of the words in the text. The latter aspect has been part of our motivation, since there are a big amount of electronic documents that could be useful for discovering sequential patterns, which often cannot be seen at first sight and could be helpful for text analysis; achieving in this manner a text mining process.

Since the number of frequent sequences can be huge, it is possible considering only those which are not contained in another frequent sequence, it means, the Maximal Frequent Sequences (MFS's) can be considered as a compact representation of all frequent sequences. When a MFS is found, the *words*, *length* and *frequency* of such MFS are determined by the text. The last characteristic is very important because the frequency allows having support for that MFS. Other important feature of the MFS's is that they can be extracted independently of the language. Frequent sequences preserve, in some way, the natural sequential order of the text. Moreover, by its legibility, MFS's are easy to understand by humans.

The above mentioned characteristics make MFS's suitable to be applied in specific text mining tasks like document clustering and classification; or in tasks like information retrieval, question answering, text summarization, etc. However, the MFS discovering problem has received special attention due to the big amount of combinations that have to be reviewed for discovering such patterns. For example, if a frequent sequence has 100 elements then $2^{100} - 1 \approx 10^{30}$ combinations have to be reviewed for extracting such pattern. This problem has been classified as NP-hard.

This dissertation deals with the problem of discovering MFS's in text. It is important to remark, that the main objective of this dissertation was to propose algorithms for improving the search of MFS's from textual information. However, the proposed algorithms can work over any other kind of sequential information like DNA sequences, WEB logs, etc.; it is, with objects that describe a sequential behavior through symbols.

In this document we present the development of four algorithms for discovering all MFS's, two from a collection of documents, Dimasp-C_0 and Dimasp-C_n; and two from a single document, Dimasp-D_0 and Dimasp-D_n. The subindex in Dimasp-C and Dimasp-D algorithms correspond to the number of words allowed, in the text, between the words that form a frequent sequence; with this parameter it is possible to preserve the context of the words of each MFS. The new algorithms follow the "pattern-growth strategy" where small frequent sequences are finding first with the objective of growing them to obtain MFS's. Examples of "growing-pattern algorithms" are GSP and MineMFS (Ahonen), the latter was specially designed for text; however, MineMFS does not find all the MFS's. Other search strategy is the "candidate subsequence generation" in which from small frequent sequences, a new set of potentially frequent sequences of greater size is proposed and each one of them is verified. This process is repeated until no more frequent sequences can be obtained. Examples of "candidate subsequence generation algorithms" are GenPrefixSpan, Delisp and cSPADE.

Using the Reuters-21578 collection, Dimasp-C_0 and Dimasp-C_n algorithms were compared against GSP, GenPrefixSpan, Delisp and cSPADE algorithms. According to the experiments, Dimasp-C_0 and Dimasp-C_n outperform GSP, GenPrefixSpan, Delisp and cSPADE algorithms. Also it is possible to see that Dimasp-C_0 outperforms Dimasp-C_n when n=0.

For searching MFS's in a single document, first we had to define the problem, since there are important differences when a single document, instead of a collection of documents, is considered. The Alex collection, which has big documents, was used for testing Dimasp-D_0 and Dimasp-D_n. No comparison was done because this is a new problem and there is not other algorithm to compare with.

Resumen

La información de un documento de texto la encontramos descrita de manera secuencial mediante palabras. Desde este punto de vista, el conocimiento transmitido por el autor de un texto es estructurado de manera secuencial. Así, el texto no sólo sirve para el fin que determinó el autor originalmente, sino que también es posible descubrir conocimiento nuevo a partir del orden secuencial de las palabras que frecuentemente se presenta en un texto. Este último aspecto ha sido precisamente parte de la motivación de esta disertación, pues existe una gran cantidad de documentos electrónicos disponibles que permitirían descubrir patrones en el texto que difícilmente pueden determinarse a primera vista y que pueden ser de gran utilidad para el análisis de texto; realizando de esta manera un proceso de *minería* sobre el texto.

Debido a que el número de secuencias frecuentes SF's encontradas puede ser enorme, es posible considerar únicamente aquellas SF's que no están contenidas dentro de otras, es decir, utilizar las *secuencias frecuentes maximales* (SFM's) como la presentación compacta del conjunto de SF's. Cuando se descubre una SFM, es el contenido del texto el que determina las *palabras, longitud y frecuencia* de la SFM. Esta última característica es muy importante porque la *frecuencia* nos permite tener un soporte sobre la existencia de dicha SFM. Otra de las características importantes de las SFM's radica en su extracción de manera *independiente del lenguaje*, incluso de texto que no esté bien escrito. Las SF's de texto preservan, en cierto modo, la *naturaleza secuencial del texto*. Incluso, por su legibilidad, las SFM's son *entendibles por el humano*.

Por las características mencionadas anteriormente, las SFM's son potencialmente útiles para tareas específicas de la minería de texto como la clasificación y agrupamiento de documentos; en el análisis automático de texto; en la recuperación de información; en la búsqueda de respuestas; extracción de hipónimos y en la elaboración de resúmenes, entre otras. Sin embargo, el problema de descubrimiento de SFM's ha requerido de atención especial debido al gran número de combinaciones que se tienen que revisar en su extracción. Por ejemplo, si una SF tiene una longitud de 100 elementos se tendrían que revisar $2^{100} - 1 \approx 10^{30}$ combinaciones antes de poder extraer dicho patrón. Este problema ha sido clasificado como un problema NP-difícil.

La presente disertación aborda el problema del descubrimiento de SFM's en texto. Cabe señalar, que el objetivo principal de la tesis es proponer algoritmos que permitan mejorar la búsqueda de las SFM's a partir de información textual. Sin embargo, los algoritmos desarrollados en esta tesis pueden trabajar con otro tipo de información como secuencias de ADN, registros de visitas de páginas WEB, etc. Es decir, con objetos que se describan de manera secuencial mediante símbolos.

En este documento se presenta el desarrollo de cuatro algoritmos para el descubrimiento de SFM's en texto partiendo de una colección de documentos, Dimasp-C_0 y Dimasp-C_n; y partiendo de un sólo documento Dimasp-D_0 y Dimasp-D_n. El subíndice tanto en los algoritmos Dimasp-C como Dimasp-D corresponde al número de palabras permitidas en el texto entre las palabras que forman una secuencia frecuente; mediante este parámetro es posible preservar el contexto de las palabras de las SFM's. Los algoritmos desarrollados emplean la estrategia de búsqueda de "crecimiento de patrones" en la cual se encuentran primero secuencias frecuentes pequeñas que después se hacen crecer con el objetivo de encontrar las maximales. Algoritmos del tipo crecimiento de patrones con restricción GAP son GSP y MineMFS (propuesto por Ahonen), este último fue diseñado especialmente para texto, sin embargo MineMFS no encuentra todas las SFM's. Otra estrategia de búsqueda es la conocida como "generación de secuencias candidatas", en la cual se generan combinaciones de secuencias frecuentes pequeñas con el objetivo de proponer posibles secuencias frecuentes de mayor tamaño. Algoritmos del tipo "generación de secuencias candidatas" son GenPrefixSpan, Delisp y cSPADE.

Dimasp-C_0 y Dimasp-C_n se compararon de manera experimental con los algoritmos GSP, GenPrefixSpan, Delisp y cSPADE, utilizando la colección de noticias en inglés Reuters-21578. De acuerdo a los experimentos realizados se observa que Dimasp-C_0 y Dimasp-C_n mejoran la búsqueda de SFM's en texto.

Para el problema de búsqueda de SFM's para un sólo documento, el problema tuvo que ser definido primero, pues existen diferencias importantes cuando se parte de un documento o de una colección de documentos. Los experimentos realizados con Dimasp-D_0 y Dimasp-D_n utilizaron documentos de la colección Alex, la cual tiene documentos grandes. En este caso, no se realizaron comparaciones porque se trata de un nuevo problema y no hay otro algoritmo con el cual compararse.

Contenido

Página

Lista de figuras

Lista de tablas

Artículos publicados

Publicaciones de los algoritmos desarrollados en esta tesis:

- René A. García-Hernández, José Fco. Martínez-Trinidad and Jesús Ariel Carrasco-Ochoa, *A Fast Algorithm to Find All the Maximal Frequent Sequences in a Text*, 9th Iberoamerican Congress on Pattern Recognition (CIARP'2004), Lecture Notes in Computer Science, vol. 3287, Springer-Verlag 2004. pp. 478-486.
- René A. García-Hernández, José Fco. Martínez-Trinidad and Jesús Ariel Carrasco-Ochoa, *A New Algorithm for Fast Discovery of Maximal Sequential Patterns in a Document Collection*, 7th Intelligent Text Processing and Computational Linguistics (CICLing'2006), Lecture Notes in Computer Science, vol. 3878, Springer-Verlag 2006. pp. 514-523.

Publicaciones donde fueron aplicados algunos de los algoritmos desarrollados en esta tesis:

- Osslan O. Vergara-Villegas, René A. García-Hernández, J. Ariel Carrasco-Ochoa, Raúl Pinto Elías, José F. Martínez-Trinidad. "*Data Preprocessing by Sequential Pattern Mining for LZW*", 6th Mexican International Conference on Computer Science (ENC 2005), ISBN 0-7695-2454-0, IEEE Computer Society Press, September 2005. pp. 82–87
- Hernández-Reyes Edith, García-Hernández Rene A., Carrasco-Ochoa J. A., Martínez-Trinidad J. Fco. "*Document Clustering based on Maximal Frequent Sequences*", 5th International Conference on NLP (Fintal 2006), Tapio Salakoski et al. (Eds.), ISBN 3-540-37334-9, Lecture Notes in Computer Science Vol. 4139, Springer-Verlag, Turku, Finland, August 2006, pp. 257-267.
- Claudia Denicia-Carral, Manuel Montes-y-Gómez, Luis Villaseñor-Pineda, René García Hernández. "*A Text Mining Approach for Definition Question Answering*", 5th International Conference on NLP (Fintal 2006), ISBN 978-3-540-37334-6, Lecture Notes in Artificial Intelligence, Vol. 4139, Springer-Verlag 2006, pp. 76–86.
- Hernández-Reyes Edith, Carrasco-Ochoa J. A., Martínez-Trinidad J. Fco. García-Hernández Rene A. "*Document Representation Based on Maximal Frequent Sequence Sets*". 11th Iberoamerican Congress on Pattern Recognition (CIARP 2006), ISBN 3-540-46556-1, Lecture Notes in Computer Science, Vol. 4225, Springer-Verlag, Cancún, México, 2006, pp. 854–863.

CAPÍTULO 1

Introducción

Durante las últimas décadas se ha presentado un crecimiento en la cantidad de información almacenada en medios electrónicos. Se estima que el 20 por ciento de la información electrónica de una empresa está almacenada en bases de datos [Leavitt 2002], donde es posible acceder de manera fácil a los objetos y registros de dicha información. Debido al crecimiento de la información en bases de datos y a los beneficios que se pueden obtener a partir de ésta; ha cobrado gran interés el análisis de la información contenida en bases de datos. Interés que ha sido abordado por el área de investigación "*Descubrimiento de Conocimiento en Bases de Datos*", la cual ha sido definida por Fayyad [Fayyad 1996] como:

> *"El descubrimiento de conocimiento en bases de datos es el proceso no trivial de identificar patrones en datos los cuales deben de ser válidos, novedosos, potencialmente útiles y entendibles."*

Es decir, el objetivo de esta área consiste en identificar patrones en bases de datos que sean válidos, novedosos, potencialmente útiles y entendibles. Como *patrones válidos* se esperan

patrones que tengan un alto grado de confianza o soporte. Por *patrones novedosos* y *potencialmente útiles*, se entiende que sean relevantes, interesantes y que nos permitan descubrir conocimiento a partir de ellos. Finalmente, se busca que sean *patrones entendibles* por el ser humano. El paso clave [Fayyad 1996] en el proceso de descubrimiento de conocimiento en bases de datos es la *Minería de Datos*, la cual siguiendo a Fayyad está definida como:

> *"La minería de datos es un paso en el proceso de descubrimiento de conocimiento en datos que consiste en la aplicación de algoritmos de análisis y descubrimiento de datos que, bajo las limitaciones de eficiencia computacional, producen una enumeración particular de patrones acerca de los datos"*

Por lo tanto, la minería de datos está enfocada al análisis de datos y al desarrollo de algoritmos de descubrimiento, sin olvidar las limitaciones computacionales de tiempo y espacio requerido por las computadoras.

Por otro lado, se estima que el 80 por ciento restante de la información electrónica de una empresa está almacenada de manera textual [Leavitt 2002]. En este caso, el interés por obtener patrones en texto que sean válidos, novedosos, potencialmente útiles y entendibles por el humano motivó el surgimiento del área de *Descubrimiento de Conocimiento en Texto* Kodratoff 1999][Feldman 1995][Karanikas 2002]. No obstante, el paso clave en este proceso es la *Minería de Texto*, la cual, parafraseando la minería de datos [Karanikas 2002][Montes 2002], está definida como:

> *"La minería de texto es un paso en el proceso de descubrimiento de conocimiento en texto que consiste en la aplicación de algoritmos de análisis y descubrimiento en texto que, bajo las limitaciones de eficiencia computacional, producen una enumeración particular de patrones acerca del texto"*

El desarrollo de los algoritmos de minería de texto requiere atención especial puesto que van a ser aplicados para analizar grandes volúmenes de información. Sin embargo, en la minería de texto es necesario desarrollar algoritmos adecuados al texto que permitan obtener una enumeración de patrones a partir de este tipo de información.

Dado que, uno de los problemas que ha surgido en la minería de datos es el análisis de la información que mantiene un orden en sus registros simbólicos a través del tiempo, es decir, de la información descrita de manera secuencial. Este problema ha sido abordado por el área de investigación denominada *Minería de Patrones Secuenciales*, en donde el objetivo

principal es extraer datos que se presentan frecuentemente, pero preservando su orden secuencial dentro de la información, los cuales se llaman ***patrones secuenciales***. Puesto que el texto mantiene un orden secuencial de las palabras entonces también es posible analizar y descubrir patrones válidos, novedosos, potencialmente útiles y entendibles por un humano a partir de información textual, los cuales a su vez podrían ser de gran utilidad, por ejemplo, para descubrir conocimiento a partir de esta fuente de información.

En la Minería de Patrones Secuenciales, una ***secuencia de palabras se considera frecuente*** si ésta se encuentra en al menos un cierto número de documentos (umbral mínimo de frecuencia). Cabe señalar que a una secuencia frecuente se le conoce también como un ***patrón secuencial***. Por ejemplo, a partir de la colección de documentos de la tabla 1.1 se obtienen como secuencias frecuentes, con un umbral de frecuencia igual a 3, las secuencias mostradas en la tabla 1.2.

Tabla 1.1 Ejemplo de una colección con cinco documentos de texto.

1.	*THE PRESIDENT OF THE UNITED STATES IS A BAD PRESIDENT*
2.	*THE PRESIDENT GEORGE BUSH IS A BAD PRESIDENT*
3.	*THE PRESIDENT GEORGE W. BUSH IS A GOOD PRESIDENT*
4.	*THE PRESIDENT BUSH WAS A BAD PRESIDENT*
5.	*THE PRESIDENT BUSH IS A REALLY BAD PRESIDENT*

Como se observa en la tabla 1.2, el descubrimiento únicamente de secuencias frecuentes no tendría mucha utilidad en el análisis de texto, ya que a partir de cinco documentos pequeños se formaron 109 secuencias frecuentes. Una forma de reducir la representación del conjunto de secuencias frecuentes consiste en tomar en cuenta sólo aquellas que no son subsecuencia de alguna otra, *i.e.* que son ***maximales***. De esta manera, de la colección de cinco documentos de la tabla 1.1 se encuentran sólo tres secuencias frecuentes maximales (SFM's) las cuales están subrayadas en la de la tabla 1.2. En este sentido, una de las propiedades de las SFM's es que todas aquellas subsecuencias que pueden formarse a partir de una SFM son también frecuentes. En otras palabras, una SFM contiene o representa a todas sus subsecuencias frecuentes; obteniendo así una ***representación compacta*** de todas las subsecuencias que se pueden formar a partir de dicha SFM.

Aunque para este ejemplo las tres SFM's tiene coherencia al momento de leerse, en la práctica pudiera no ser así, ya que no se tiene una restricción de la separación máxima, en el texto, entre las palabras que forman una secuencia frecuente. A dicha restricción en la

formación de patrones frecuentes se le ha llamado *restricción GAP*. De esta manera, para valores pequeños de GAP se encontrarían patrones más legibles donde el contexto de las palabras que forman una SFM se mantenga. Por ejemplo, las SFM's encontradas con un umbral de frecuencia de 3 y un GAP igual a 0 (es decir, SFM's con palabras adyacentes o consecutivas) son mostradas en la tabla 1.3.

Tabla 1.2 Secuencias frecuentes extraídas a partir de los documentos de la tabla 1.1 con un umbral de frecuencia igual a 3.

DE TAMAÑO 1
1. A
2. BAD
3. IS
4. PRESIDENT
5. THE
6. BUSH

DE TAMAÑO 2
7. THE,PRESIDENT
8. THE,IS
9. THE,A
10. THE,BAD
11. PRESIDENT,IS
12. PRESIDENT,A
13. PRESIDENT,BAD
14. PRESIDENT,PRESIDENT
15. IS,A
16. IS,BAD
17. IS,PRESIDENT
18. A,BAD
19. A,PRESIDENT
20. BAD,PRESIDENT
21. THE,BUSH
22. PRESIDENT,BUSH
23. BUSH,IS
24. BUSH,A
25. BUSH,BAD
26. BUSH,PRESIDENT

DE TAMAÑO 3
27. THE,PRESIDENT,IS
28. THE,PRESIDENT,A
29. THE,PRESIDENT,BAD
30. THE,PRESIDENT,BUSH
31. THE,IS,A
32. THE,IS,BAD
33. THE,IS,PRESIDENT
34. THE,A,BAD
35. THE,A,PRESIDENT
36. THE,BAD,PRESIDENT
37. THE,PRESIDENT,PRESIDENT
38. PRESIDENT,IS,A
39. PRESIDENT,IS,BAD
40. PRESIDENT,IS,PRESIDENT
41. PRESIDENT,A,BAD
42. PRESIDENT,A,PRESIDENT
43. PRESIDENT,BAD,PRESIDENT
44. IS,A,BAD
45. IS,A,PRESIDENT
46. IS,BAD,PRESIDENT
47. A,BAD,PRESIDENT
48. THE,BUSH,IS
49. THE,BUSH,A
50. THE,BUSH,BAD
51. THE,BUSH,PRESIDENT
52. PRESIDENT,BUSH,IS
53. PRESIDENT,BUSH,A
54. PRESIDENT,BUSH,BAD
55. PRESIDENT,BUSH,PRESIDENT
56. BUSH,IS,A
57. BUSH,IS,PRESIDENT
58. BUSH,A,BAD
59. BUSH,A,PRESIDENT
60. BUSH,BAD,PRESIDENT

DE TAMAÑO 4
61. THE,PRESIDENT,IS,A
62. THE,PRESIDENT,IS,BAD
63. THE,PRESIDENT,A,BAD
64. THE,PRESIDENT,A,PRESIDENT
65. THE,PRESIDENT,BUSH,IS
66. THE,PRESIDENT,BUSH,A
67. THE,PRESIDENT,BUSH,BAD
68. THE,IS,A,BAD
69. THE,IS,A,PRESIDENT
70. THE,IS,BAD,PRESIDENT
71. THE,A,BAD,PRESIDENT
72. PRESIDENT,IS,A,BAD
73. THE,PRESIDENT,IS,PRESIDENT
74. THE,PRESIDENT,BAD,PRESIDENT
75. THE,PRESIDENT,BUSH,PRESIDENT
76. PRESIDENT,IS,A,PRESIDENT
77. PRESIDENT,IS,BAD,PRESIDENT
78. PRESIDENT,A,BAD,PRESIDENT
79. IS,A,BAD,PRESIDENT
80. THE,BUSH,IS,A
81. THE,BUSH,IS,PRESIDENT
82. THE,BUSH,A,BAD
83. THE,BUSH,A,PRESIDENT
84. THE,BUSH,BAD,PRESIDENT
85. PRESIDENT,BUSH,IS,A
86. PRESIDENT,BUSH,IS,PRESIDENT
87. PRESIDENT,BUSH,A,BAD
88. PRESIDENT,BUSH,A,PRESIDENT
89. PRESIDENT,BUSH,BAD,PRESIDENT
90. BUSH,IS,A,PRESIDENT
91. BUSH,A,BAD,PRESIDENT

DE TAMAÑO 5
92. THE,PRESIDENT,IS,A,BAD
93. THE,PRESIDENT,IS,A,PRESIDENT
94. THE,PRESIDENT,IS,BAD,PRESIDENT
95. THE,PRESIDENT,A,BAD,PRESIDENT
96. THE,PRESIDENT,BUSH,IS,A
97. THE,PRESIDENT,BUSH,IS,PRESIDENT
98. THE,PRESIDENT,BUSH,A,BAD
99. THE,PRESIDENT,BUSH,A,PRESIDENT
100. THE,PRESIDENT,BUSH,BAD,PRESIDENT
101. THE,IS,A,BAD,PRESIDENT
102. PRESIDENT,IS,A,BAD,PRESIDENT
103. THE,BUSH,IS,A,PRESIDENT
104. THE,BUSH,A,BAD,PRESIDENT
105. PRESIDENT,BUSH,IS,A,PRESIDENT
106. PRESIDENT,BUSH,A,BAD,PRESIDENT

DE TAMAÑO 6
107. *THE,PRESIDENT,IS,A,BAD,PRESIDENT*
108. *THE,PRESIDENT,BUSH,IS,A,PRESIDENT*
109. *THE,PRESIDENT,BUSH,A,BAD,PRESIDENT*

Tabla 1.3 SFM's de los documentos de la tabla 1.1, con un umbral igual a 3 y una restricción GAP=0.

1. *THE PRESIDENT*
2. *BUSH IS A*
3. *A BAD PRESIDENT*

Otra posibilidad es extraer SFM's con umbral de 3 y restricción GAP de 1; lo que produciría las SFM's presentadas en la tabla 1.4.

Tabla 1.4 SFM's de los documentos de la tabla 1.1, con un umbral igual a 3 y una restricción GAP=1.

1. *IS A PRESIDENT*
2. *BUSH IS A*
3. *BUSH A PRESIDENT*
4. *IS A BAD PRESIDENT*
5. *THE PRESIDENT BUSH A BAD PRESIDENT*

Como se aprecia en las tablas 1.3 y 1.4, las SFM's tienden a ser de mayor longitud cuando la restricción de separación (GAP) entre las palabras aumenta. Por ejemplo, la quinta SFM de la tabla 1.4 fue obtenida de los documentos dos, cuatro y cinco; usando las palabras de la colección original que se muestran subrayadas en la tabla 1.5. Como se aprecia en este ejemplo, puede existir una separación de una palabra (GAP=1) entre las palabras que forman la secuencia.

Tabla 1.5 Palabras de los documentos de la tabla 1.1 que son utilizados para obtener la quinta SFM de la tabla 1.4.

1. *THE PRESIDENT OF THE UNITED STATES IS A BAD PRESIDENT*
2. *THE PRESIDENT GEORGE BUSH IS A BAD PRESIDENT*
3. *THE PRESIDENT GEORGE W. BUSH IS A GOOD PRESIDENT*
4. *THE PRESIDENT BUSH WAS A BAD PRESIDENT*
5. *THE PRESIDENT BUSH IS A REALLY BAD PRESIDENT*

Como se observa en los ejemplos anteriores, las palabras de las secuencias frecuentes preservan su contexto y ***naturaleza secuencial***. Además, es el contenido de los documentos lo que determina las palabras, secuencia, frecuencia y longitud de una SFM. Otra de las características importantes de las SFM's reside en su extracción de manera ***independiente del lenguaje***, incluso de texto que no esté bien escrito como sucede con los documentos de

la tabla 1.1. Además, cuando es aplicada una restricción GAP con valores pequeños [0,1,2], las SFM's extraídas con restricción GAP conservan, en cierto modo, el *contexto original de sus palabras*.

Por otro lado, si consideramos a un documento como una secuencia de palabras entonces resultaría provechoso poder realizar la extracción de secuencias frecuentes maximales considerando nada más la información de un sólo documento. En este sentido, al tener una secuencia tan grande, como lo puede ser un documento, existiría también un conjunto importante de SFM's que pueden extraerse a partir de dicho documento. Sin embargo, este problema presenta características que lo hacen diferente del normalmente definido en minería de patrones secuenciales. Por ejemplo, cuando se hace la búsqueda de SFM's en una colección de documentos es la misma colección quien determina, por medio de cada documento, los límites posibles donde puede ocurrir una, y sólo una, de las secuencias que forman la SFM. En cambio, el problema de búsqueda de SFM's considerando solamente un documento es aun más complicado porque no se conocen los límites *a priori* de cada una de las secuencias que forman el patrón secuencial. Es decir, se tendrían que conocer, para cada secuencia frecuente, las posiciones donde ésta puede empezar y terminar, lo cual va a depender por supuesto del umbral de frecuencia. Además, se debe tener en cuenta que los límites a considerar tienen que permitir encontrar la secuencia frecuente más larga (maximal) de acuerdo al umbral de frecuencia. Algunos de estos problemas se ejemplifican más adelante en el capítulo cuatro.

No obstante, la dificultad de extraer todas las SFM's reside en los recursos computacionales que son necesarios para su obtención, es decir, el tiempo y espacio de almacenamiento requerido. Por lo tanto, el problema fundamental en esta tesis es encontrar secuencias frecuentes maximales de manera más rápida a partir de información textual considerando la restricción GAP. Así, el objetivo principal es desarrollar algoritmos de minería de patrones secuenciales en información textual, por lo cual se deben desarrollar dos tipos de algoritmos, uno para realizar la extracción de SFM's a partir de una colección de documentos considerando la restricción GAP y otro para permitir la extracción de SFM's considerando un sólo documento con restricción GAP. Cabe señalar que en el último caso fue necesario definir el problema porque en la literatura relacionada sólo existen algoritmos para la búsqueda de SFM's con restricción GAP dada una colección de secuencias.

De esta manera se espera contribuir con herramientas de minería de texto independientemente del lenguaje que nos permitan realizar un análisis de información textual con el objetivo de obtener patrones válidos, novedosos, potencialmente útiles y

finalmente entendibles por el humano. En este sentido, estas herramientas serían un primer paso para llegar al fin último de descubrimiento de conocimiento. Por ejemplo, las SFM's han sido utilizadas en tareas propias del procesamiento automático de texto [Ahonen 1997], descubrimiento de conocimiento [Ahonen 2002] y en tareas específicas de Minería de Texto como lo son: agrupamiento de documentos [Doucet 2003][Jeong 2005, 2005a], determinación del autor de un texto [Coyotl 2006] y detección de tópicos [Yap 2006]. Por sus características, las SFM's son atractivas para ser utilizadas en extracción de frases clave [Frank 1999][Turney 1997, 1999, 199a], clasificación y navegación de colecciones de documentos. Cabe señalar que las SFM's ya son utilizadas en tareas de procesamiento automático de texto como lo es: recuperación de información [Doucet 2004, 2004a][Jeong 2005a], extracción de hipónimos en la WEB [Ortega 2007, 2007a], búsqueda de respuestas en un sólo lenguaje [Denicia 2006, 2007] o de manera multilingüe [Marina 2007]. Sin embargo las SFM's también serían atractivas para extracción de información, resúmenes de documentos o desambiguación del sentido de una palabra, entre otras. No obstante, las SFM's serían también potencialmente útiles para resolver problemas como la compresión de datos [Osslan 2005], en el análisis de secuencias de ADN [Hajime 2002]; en la detección de intrusos en una red.

1.1 Organización de la tesis

En este capítulo se introdujeron los antecedentes de esta disertación. En el capítulo dos se exponen los conceptos sobre minería de patrones frecuentes junto con su estado del arte, lo cual permite establecer el problema de investigación, los objetivos y justificación de la investigación. En el capítulo tres se presenta la definición del problema y los algoritmos para la búsqueda de SFM's en una colección de documentos con GAP=0 y GAP=n; además son presentados los experimentos y comparaciones con algoritmos de este tipo. En el capítulo cuatro se presenta la definición del problema y los algoritmos para la búsqueda de SFM's en un sólo documento con GAP=0 y GAP=n. Además, se muestra el comportamiento de los algoritmos desarrollados cuando son aplicados en documentos de gran tamaño. Por último, en el capítulo cinco se presentan las conclusiones, aportaciones y trabajo futuro derivados de esta disertación.

1.2. Sumario

Inicialmente en este capítulo se introdujeron conceptos básicos como descubrimiento de conocimiento en bases de datos, descubrimiento de conocimiento en texto, minería de datos, minería de texto y minería de patrones secuenciales. También, se mencionó lo que en minería de patrones secuenciales se entiende por secuencia, secuencia frecuente, patrón secuencial y secuencia frecuente maximal. En este capítulo se muestra con algunos ejemplos sencillos, cómo la minería de patrones secuenciales permitiría analizar documentos de texto cuando la restricción GAP es contemplada. Los ejemplos mostrados también permitieron ilustrar algunas de las características principales de las SFM's como su independencia del lenguaje, su representación compacta y se preservación tanto del orden secuencial como del contexto original de las palabras; sin olvidar que es el contenido de los documentos quien determina las palabras, secuencia, frecuencia y longitud de una SFM descubierta. También se introdujeron brevemente el problema y objetivos de esta tesis. Por último se proporcionó la organización de esta tesis.

CAPÍTULO 2

Minería de Patrones Secuenciales

En este capítulo se aborda primero la *minería de patrones frecuentes*. En específico, se presenta la *minería de patrones frecuentes en conjuntos de datos* como un preámbulo a la de *minería de patrones frecuentes en secuencias de datos*. Esto es necesario ya que el problema de minería de patrones sobre conjuntos de datos es el precursor de la minería de secuencias de conjuntos de datos, incluso algunos algoritmos de minería de patrones frecuentes que trabajaban sobre conjuntos evolucionaron y fueron adaptados para la minería de secuencias. Después se mencionan las clasificaciones propuestas para los algoritmos de minería de conjuntos frecuentes y para los algoritmos de patrones secuenciales. Posteriormente, en la sección de estado del arte se describen los trabajos relacionados más relevantes, incluyendo los árboles de sufijos. Por último se establece el problema, objetivos y motivación de la presente disertación en el desarrollo de los algoritmos de descubrimiento de patrones secuenciales maximales.

2.1 Minería de Patrones Frecuentes Maximales

Uno de los problemas fundamentales de la minería de datos es la minería de patrones frecuentes maximales [Yang 2004], la cual se ha llevado a cabo sobre diferentes tipos de datos como conjuntos, secuencias, árboles y grafos.

Con el fin de entender el tipo y origen de los algoritmos desarrollados para la *minería de patrones secuenciales maximales* se describe primero la *minería de conjuntos frecuentes* (*Frequent Itemset Mining*).

2.1.1 Minería de Conjuntos Frecuentes

En general, el problema de minería de conjuntos frecuentes fue introducido por Agrawal [Agrawal 1993], el cual está formulado como:

> *"Dada una colección de conjuntos, el problema consiste en descubrir todos los subconjuntos tales que sus elementos estén contenidos, en al menos un cierto número de conjuntos (determinado con un umbral de frecuencia), en la colección original de conjuntos."*

Entre los conjuntos frecuentes descubiertos están los *conjuntos frecuentes maximales*, los cuales permiten representar a todos los conjuntos frecuentes, puesto que todos los subconjuntos que se puedan formar, a partir de un conjunto frecuente maximal, son también subconjuntos frecuentes. Por tal motivo, una forma de reducir el número de conjuntos frecuentes es encontrar solamente los conjuntos frecuentes maximales.

Por ejemplo, para el problema de la *canasta de supermercado* se tiene una base de datos como la de la tabla 2.1a, donde hay cinco conjuntos de productos los cuales fueron comprados por un cliente en diferentes transacciones. La tabla 2.1b muestra los conjuntos frecuentes maximales que existen con un umbral de frecuencia igual a 2. Como se aprecia en los conjuntos frecuentes maximales, el orden en que fueron comprados los productos por el cliente es irrelevante para este tipo de descubrimiento. Por ejemplo, aunque en las transacciones T4 y T5 de la tabla 2.1, los productos *"Leche"* y *"Cerveza"* aparecen en orden diferente (de izquierda a derecha) forman el mismo subconjunto frecuente maximal.

En [Yang 2004] se demuestra que este problema es NP-difícil. Por ejemplo, para obtener dichos patrones frecuentes los algoritmos tienen que revisar explícitamente 2^{m-1} conjuntos de elementos [Dao 1998], donde m es el número de elementos en el conjunto maximal. De esta manera, para encontrar un conjunto maximal de tamaño 100, se tendrían que revisar 2^{100}-1 (aproximadamente 10^{30}) conjuntos. En algunos algoritmos [Agrawal 1994a], esta revisión es explícita porque incluyen pasos para recorrer la base de datos con el fin de verificar la frecuencia de cada conjunto.

Tabla 2.1 Ejemplo de la "canasta de supermercado". a) Base de datos de transacciones por cliente, b) Conjuntos frecuentes maximales con umbral igual a 2.

a)

Transacción	Productos
T1	{Mantequilla, Pan, Mermelada}
T2	{Pan, Mantequilla, Cerveza}
T3	{Pan, Leche, Mantequilla}
T4	{Cerveza, Leche, Mermelada, Pan}
T5	{Leche, Cerveza}

b)

Conjuntos frecuentes maximales
{Pan, Mermelada}
{Cerveza, Leche}
{Leche, Pan}
{Cerveza, Pan}
{Mantequilla, Pan}

Este tipo de descubrimiento también ha sido aplicado a textos como en [Feldman 1996][Ahonen 1999a], donde los elementos son palabras o frases del texto. Esto se hace con el objetivo de encontrar co-ocurrencias de palabras en una colección de documentos en donde no es importante el orden ni la separación que las palabras puedan tener dentro del documento. Así, los problemas, en los que es importante el orden de aparición de los elementos, han motivado el surgimiento del área de minería de patrones secuenciales.

2.1.2 Minería de Patrones Secuenciales

De manera similar al problema de minería de conjuntos frecuentes, se ha definido el problema de encontrar ***secuencias frecuentes*** a partir de una colección de secuencias. Este problema fue introducido por Agrawal y Srikant [Agrawal 1994], que en general está formulado como:

> *"Dado un conjunto de secuencias, donde cada secuencia consiste de una lista de elementos y cada elemento consiste de un conjunto de ítems; y un umbral de soporte mínimo especificado por un usuario, el problema consiste en encontrar todas las secuencias frecuentes, i.e. secuencias las cuales su frecuencia de ocurrencia en el conjunto de secuencias no sea menor que el soporte mínimo."*

Como en el problema de minería de conjuntos frecuentes, al descubrir todas las secuencias frecuentes (SF's) se descubren también las secuencias frecuentes maximales (SFM's), las cuales no están contenidas en alguna otra secuencia frecuente. A las secuencias frecuentes se le conoce también como patrones secuenciales (PS's).

Con el objetivo de ilustrar los tipos de secuencias frecuentes encontradas en una base de datos (BD) de transacciones de un negocio de rentas de videos, supongamos una BD como la tabla 2.2a donde se tiene registro de las compras de cinco clientes (identificados por Id), el tiempo en que se hizo la renta del video y los nombres de los videos rentados. La base de datos de la tabla 2.2a puede ser ordenada de manera secuencial de acuerdo al tiempo en que se efectuaron las transacciones, lo cual se muestra en la tabla 2.2b. Las SFM's que existen en la tabla 2.2b con una frecuencia de 2 se muestran en la tabla 2.2c, las cuales permiten conocer el comportamiento de un cliente en las rentas de videos. De esta manera, los conjuntos encontrados en la tabla 2.2c permiten saber que si un cliente renta los videos *"La comunidad del anillo"* y *"Las dos torres"* entonces es probable que la siguiente vez rente el video *"El retorno del rey"*, probabilidad soportada por 2 casos (clientes) en la BD. Es por eso que el umbral de frecuencia requerido es también conocido como el soporte requerido.

Como se aprecia en la tabla 2.2c, a diferencia de los conjuntos frecuentes maximales, las SFM's encontradas tienen un orden secuencial. Por ejemplo, los videos *"El padrino"* y *"La guerra de las galaxias"* (que fueron rentados por los clientes 2 y 3) no constituyen una SFM por estar en diferente orden secuencial.

El tipo de secuencias extraídas describen en sí el comportamiento de un objeto. Por ello, las secuencias frecuentes han tenido diversas aplicaciones, por ejemplo en:

- Detección de tipos de hepatitis [Sujeevan 2005]
- Predicción en navegación de páginas WEB [Bamshad 2002]
- Visualización de las páginas WEB más consultadas [Youssefi 2004]
- Detección de eventos frecuentes en alarmas para predicción y control [Pei 2001a]

Tabla 2.2 Ejemplo de las SFM's encontradas en una base de datos de las rentas de un video

a) Base de datos de las rentas realizadas por un cliente

Id	Tiempo en que se efectuaron las rentas	Videos rentados
1	Junio 22	{Las dos torres, La comunidad del anillo, El silencio de los inocentes}
1	Junio 25	{ Hannibal }
1	Junio 26	{El retorno del rey, Dragón rojo}
2	Junio 10	{La guerra de las galaxias}
2	Junio 15	{La comunidad del anillo, Las dos torres}
2	Junio 20	{El retorno del rey, El padrino}
3	Junio 5	{El silencio de los inocentes, El padrino}
3	Junio 13	{La guerra de las galaxias, Hannibal }
3	Junio2 5	{Dragón rojo}

b) Compras ordenadas en el tiempo que se hizo la renta

Id	videos rentados
1	⟨ { Las dos torres, La comunidad del anillo, El silencio de los inocentes} { Hannibal } {El retorno del rey, Dragón rojo} ⟩
2	⟨{La guerra de las galaxias} {La comunidad del anillo, Las dos torres} {El retorno del rey, El padrino}⟩
3	⟨ {El silencio de los inocentes, El Padrino} { La guerra de las galaxias, Hannibal } {Dragón rojo} ⟩

c) SFM's con umbral igual a 2

Secuencias Frecuentes Maximales
⟨ {La comunidad del anillo, Las dos torres} {El retorno del rey} ⟩
⟨ {El silencio de las inocentes} {Hanníbal} {Dragón Rojo} ⟩

Igual que ocurre en la Minería de Conjuntos Frecuentes, en [Yang 2004] se demuestra que el problema de Minería de Patrones Secuenciales es NP-difícil. Por ejemplo, para encontrar una SFM de longitud m, también denotado como m~SFM, los algoritmos tienen que revisar explícitamente 2^m-1 combinaciones de elementos [Pei 2001]. Algoritmos de este tipo son AprioriAll [Agrawal 1994] y GSP [Agrawal 1996], en los cuales la revisión de las 2^m-1 secuencias es explícita, porque para cada una de estas secuencias se debe recorrer la base de datos con el fin de verificar su frecuencia.

2.1.3 Clasificación de los algoritmos de minería de patrones frecuentes

En los dos tipos de minería expuestos, minería de conjuntos frecuentes y minería de patrones secuenciales, los algoritmos están clasificados de acuerdo a la estrategia de búsqueda empleada para encontrar los patrones frecuentes maximales.

Si los algoritmos empiezan a encontrar patrones pequeños; y a partir de éstos los más grandes, entonces son algoritmos clasificados como *ascendentes* (en inglés, *bottom-up*), de lo contrario, si desde un principio se intenta encontrar los patrones maximales, con el objetivo de no buscar los más pequeños, entonces se les denomina *descendentes* (en inglés, *top-down*). Dentro de los algoritmos ascendentes, que son la mayoría, existen dos tipos: los llamados apriori (típicos) y los de crecimiento de patrones. Ejemplo de algoritmos ascendentes son: A priori [Agrawal 1994a], AprioriAll [Agrawal 1994], AprioriSome [Agrawal 1994], AprioriDynamicSome [Agrawal 1994], GSP [12] y Ahonen [Ahonen 1999]. Dentro de los algoritmos descendentes está SPLMiner [Seno 2002]. Un caso especial es el algoritmo Pincer-Search [Dao 1998][Dao 1999] el cual emplea ambas estrategias de búsqueda.

La mayoría de los algoritmos de minería de patrones frecuentes están clasificados como ascendentes, sin embargo la diferencia radica en cómo se hace el descubrimiento de nuevos conjuntos de patrones frecuentes. Los algoritmos clasificados como *típicos* están caracterizados por hacer el descubrimiento de todo el conjunto de patrones frecuentes de tamaño k+1 con base en el conjunto de patrones frecuentes de tamaño k, es decir el algoritmo hace el descubrimiento de nuevos patrones frecuentes utilizando conocimiento *a priori*. En este sentido, los algoritmos típicos generan un nuevo patrón candidato de tamaño k+1, a partir de dos patrones de tamaño k que tengan k–1 elementos iguales, lo cual va a permitir generar un nuevo *patrón candidato*, esto es llamado el *paso de generación de patrones candidatos*. Hasta este punto se sabe que el patrón frecuente sólo es candidato, lo cual hace necesario obtener su frecuencia en la base de datos con el propósito de comprobar la existencia de dicho patrón, lo cual es llamado el paso *de recorrido o escaneo de la base de datos*. Con el objetivo de evitar el paso de recorrido en la base datos, algoritmos como GSP [Agrawal 1996] realiza el *paso de poda de patrones secuenciales candidatos de tamaño* k+1, verificando la frecuencia de sus subsecuencias de tamaño k. No obstante, el paso de poda es innecesario en secuencias donde sus elementos sólo contienen un elemento, como es el caso de secuencias de palabras (texto). Una de las características importantes de estos algoritmos radica en que aquellas secuencias de longitud k las cuales no generaron secuencias válidas son entonces SFM's. Es decir, el descubrimiento de SFM's se

hace de menor a mayor longitud. De esta manera, los algoritmos clasificados como típicos tienen la característica de emplear los pasos de unión, poda y recorrido de la base de datos, lo cual es llamado *"el paso de generación de patrones candidatos"*.

Los algoritmos clasificados de tipo ***crecimiento de patrones*** como FreeSpan [Han 2000], PrefixSpan [Pei 2004][Pei 2001], SPADE [Mohammed 2000], GenPrefixSpan [Antunes 2003] y MEMISP [Ming 2002a], se caracterizan por no generar patrones candidatos y en su lugar generan las primeras secuencias frecuentes, a partir de las cuales se hace un crecimiento hasta obtener las SFM's [Pei 2002][Han 2000a]. Por la forma de búsqueda los algoritmos de crecimiento de patrones han mostrado ser más rápidos que los típicos cuando las SFM's son de mayor longitud.

También existe la clasificación de búsqueda en ***amplitud*** o en ***profundidad***. Los algoritmos como Apriori [Agrawal 1994a], AprioriAll [Agrawal 1994] y GSP [Agrawal 1996] son algoritmos de búsqueda en ***amplitud*** por buscar primero todas las secuencias frecuentes de tamaño k, y luego a partir de éstas buscan todas las secuencias frecuentes de tamaño $k+1$. Mientras los algoritmos de búsqueda en ***profundidad***, como el de Ahonen [Ahonen 1999], buscan primero una SFM antes de encontrar todas las secuencias frecuentes de tamaño k, esto se hace con la intención de eliminar subsecuencias candidatas a ser revisadas.

Aunque en esta sección se presentan las clasificaciones de los algoritmos de minería de patrones frecuentes, también existe la distinción entre los tipos de colecciones procesadas por los algoritmos de minería de patrones secuenciales. Por ejemplo, cuando el número de secuencias en las colecciones es considerado como grande (en miles) y la longitud de las secuencias como pequeña (menor de 20), entonces el problema de búsqueda de patrones secuenciales es considerado de tipo ***vertical***. En cambio, cuando el número de secuencias no es muy grande, pero la longitud de las secuencias es mayor a 20, el problema es considerado como ***horizontal***.

2.2. Estado del Arte

En un inicio, los primeros trabajos desarrollados atendieron al problema de minería de conjuntos frecuentes en una base de datos de transacciones. Posteriormente, algunos de estos trabajos y otros nuevos evolucionaron para resolver el problema de minería de patrones secuenciales en una base de datos de transacciones. No obstante, algunos algoritmos de

minería de patrones secuenciales no pueden ser aplicados tal cual para obtener patrones secuenciales, porque no contemplan la aplicación de la restricción GAP en sus algoritmos. A continuación se muestran algunos de los algoritmos desarrollados para la minería de patrones secuenciales.

2.2.1 Algoritmo GSP

El primer algoritmo propuesto para encontrar las SFM's con restricción GAP es GSP [Agrawal 1996], el cual fue propuesto en 1996 por Agrawal y es el sucesor del algoritmo AprioriAll [Agrawal 1994]. GSP y AprioriAll permiten obtener las secuencias frecuentes (SF's), sin embargo GSP permite establecer la restricción GAP en su búsqueda. Tanto GSP como AprioriAll están clasificados como típicos por emplear los pasos de generación de secuencias candidatas, poda y recorrido de la base de datos.

El algoritmo GSP empieza extrayendo el conjunto de secuencias frecuentes de longitud 2 (2~SF's) con un soporte de frecuencia mayor o igual al especificado, esto puede implicar revisiones (explícitas) de toda la base de datos con el objeto de determinar la frecuencia de las secuencias. Con el propósito de no revisar nuevamente la base de datos para encontrar el conjunto de 3~SF's, se propone un nuevo conjunto de secuencias candidatas (SC's) a partir del conjunto de 2~SF's. Proponer un nuevo conjunto de SC's es posible gracias a la propiedad de *anti-monotonicidad* la cual dice que si una secuencia no es frecuente, ninguna de sus supersecuencias es frecuente. Para generar el conjunto de k+1~SC's se verifica que los k–1 elementos del sufijo de una secuencia frecuente A de longitud k sean iguales a los k–1 elementos del prefijo de otra secuencia frecuente B de longitud k, en cuyo caso se forma una SC con los k elementos de la secuencia A, más el último elemento de la secuencia B. Una vez generadas las SC's se aplica el paso de recorrido de la base de datos para saber cuáles de las SC's tienen una frecuencia mayor al umbral requerido y con ello poder determinar el nuevo conjunto de k+1~SF's. Todas aquellas k~SF's que no generaron una k+1~SF constituyen SFM's. Este proceso continua hasta que no sea posible generar más SC's.

Por ejemplo, considerando la colección de documentos presentada en el capítulo uno, tabla 1.1, con umbral de frecuencia de 3 y un GAP=1; GSP puede encontrar todas las SFM's. Para encontrar las SFM's, GSP primero busca en la colección (tabla 1.1) todo el conjunto de 1~SF's diferentes (palabras), las cuales se muestran en la tabla 2.3. Luego se genera el conjunto de 2~SF's diferentes considerando la separación permitida de GAP igual a 1, por

ejemplo, para el primer documento *"THE PRESIDENT OF THE UNITED STATES IS A BAD PRESIDENT"* se generan las secuencias ⟨*THE,PRESIDENT*⟩, ⟨*THE,OF*⟩, ⟨*PRESIDENT,OF*⟩, ⟨*PRESIDENT,THE*⟩, ⟨*OF,THE*⟩, ⟨*OF,UNITED*⟩, ⟨*THE,UNITED*⟩, ⟨*THE,STATES*⟩, ⟨*UNITED,STATES*⟩, ⟨*STATES,IS*⟩, ⟨*STATES,A*⟩, ⟨*IS,A*⟩, ⟨*IS,BAD*⟩, ⟨*A,BAD*⟩, ⟨*A,PRESIDENT*⟩. Sin embargo, sólo las secuencias ⟨*THE,PRESIDENT*⟩, ⟨*IS,A*⟩, ⟨*IS,BAD*⟩, ⟨*A,BAD*⟩ y ⟨*A,PRESIDENT*⟩ tienen una frecuencia mayor a 3. Este proceso se realiza para todos los documentos, produciendo las SF's de tamaño k=2 mostrados en la tabla 2.3. Después, para generar una 3~SC se verifica que el sufijo de una 2~SF sea igual al prefijo de otra 2~SF. Por ejemplo, las secuencias frecuentes ⟨*THE,PRESIDENT*⟩ y ⟨*PRESIDENT,BUSH*⟩, tiene el mismo sufijo y prefijo respectivamente, *i.e.* la palabra *"PRESIDENT"*; por lo cual es posible generar la SC = ⟨*THE,PRESIDENT,BUSH*⟩, la cual tiene que ser revisada en la colección con el fin de determinar su frecuencia. En este caso, la SC ⟨*THE,PRESIDENT,BUSH*⟩ tiene una frecuencia de 3 volviéndose una SF, por lo que las secuencias ⟨*THE,PRESIDENT*⟩ y ⟨*PRESIDENT,BUSH*⟩, no son maximales. De esta manera, al terminar de generar todas las 3~SF's se conocen las 2~SFM's, que en este caso no hay alguna. Las 3~SF's ⟨*THE,PRESIDENT,BUSH*⟩ y ⟨*PRESIDENT,BUSH,A*⟩ permiten generar la 4~SC ⟨*THE,PRESIDENT,BUSH,A*⟩, puesto que el sufijo y prefijo son iguales, *i.e. PRESIDENT,BUSH*. Este proceso continúa así hasta que no sea posible generar más SC's. La tabla 2.3 muestra las secuencias generadas por GSP.

Tabla 2.3 Conjuntos de secuencias frecuentes generadas por GSP, donde la frecuencia de la secuencia se indica entre corchetes [ ] y las secuencias candidatas que no cumplen con el umbral de frecuencia de 3 son tachados. Las secuencias en negritas son las secuenciales frecuentes maximales.

SF DE TAMAÑO 1
1. [5] THE
2. [5] PRESIDENT
3. [5] IS
4. [5] A
5. [4] BAD
6. [4] BUSH

SF DE TAMAÑO 2
1. [5] THE,PRESIDENT
2. [4] IS,A
3. [4] A,BAD
4. [4] A,PRESIDENT
5. [4] BAD,PRESIDENT
6. [3] PRESIDENT,BUSH
7. [3] BUSH,IS
8. [4] BUSH,A

SF TAMAÑO 3
1. [3] THE,PRESIDENT,BUSH
2. [3] IS,A,BAD
3. **[3] IS,A,PRESIDENT**
4. [4] A,BAD,PRESIDENT
5. ~~[2] PRESIDENT,BUSH,IS~~
6. [3] PRESIDENT,BUSH,A
7. **[3] BUSH,IS,A**
8. [3] BUSH,A,BAD
9. **[3] BUSH,A,PRESIDENT**

SF DE TAMAÑO 4
1. [3] THE,PRESIDENT,BUSH,A
2. **[3] IS,A,BAD,PRESIDENT**
3. [3] PRESIDENT,BUSH,A,BAD
4. [2] ~~PRESIDENT,BUSH,A,PRESIDENT~~
5. ~~[2] BUSH,IS,A,BAD~~
6. [2] ~~BUSH,IS,A,PRESIDENT~~
7. [3] BUSH,A,BAD,PRESIDENT

SF DE TAMAÑO 5
1. [3] THE,PRESIDENT,BUSH,A,BAD
2. [3] PRESIDENT,BUSH,A,BAD,PRESIDENT

SF DE TAMAÑO 6
1. **[3] THE,PRESIDENT,BUSH,A,BAD,PRESIDENT**

GSP presenta dos características importantes, una de ellas es la capacidad de procesar colecciones de gran tamaño puesto que sólo es necesario mantener en memoria principal la generación de una SC. Otra de las características es que en la iteración k+1 se genera todo

el conjunto de k~SFM's, esto le va permitiendo al algoritmo reducir el espacio de búsqueda en cada iteración. No obstante, también esta última característica es una de sus principales desventajas puesto que en las primeras iteraciones se generan un gran número de SC's las cuales muchas de ellas no existen en la base de datos. Esta desventaja se acentúa cuando es mayor el tamaño del alfabeto debido a que existiría un mayor número de combinaciones posibles para formar una SC. Cabe observar que para encontrar una k~SFM se tuvieron que haber realizado k recorridos de la base de datos. En evaluaciones realizadas a GSP [Antunes 2003][Antunes 2005] se ha visto que su desempeño en tiempo se ve afectado de manera negativa cuando el número de elementos y tamaño de las SFM's crece, obteniendo su mejor desempeño en bases de datos de tipo vertical. En este sentido, para el problema atacado en esta tesis GSP podría presentar problemas en su desempeño, puesto que el número de palabras y la longitud de los documentos serían de gran tamaño.

2.2.2 Algoritmo de Ahonen (MineMFS)

Algunas investigaciones posteriores a la publicación del algoritmo GSP se han enfocado en reducir y acelerar la etapa de generación de Secuencias Candidatas (SC's). Uno de ellos es el algoritmo desarrollado en 1999 por Helena Ahonen [Ahonen 1999] el cual, entre los algoritmos revisados, es el único que fue creado específicamente para secuencias de palabras (colecciones de documentos) con restricción GAP. El algoritmo de Ahonen fue presentado también en [Ahonen 1999b] el cual fue nombrado después en [Ahonen 2005] como MineMFS.

Por la forma en que trabaja, el algoritmo de Ahonen está clasificado como típico, de hecho se puede considerar una extensión al algoritmo GSP. La variante en el algoritmo de Ahonen es el desarrollo de una búsqueda en profundidad después de haber realizado el descubrimiento del conjunto de secuencias frecuentes de tamaño k. La idea básica del descubrimiento del conjunto de secuencias frecuentes de tamaño k se realiza de la misma forma que en GSP. La diferencia radica en que Ahonen elimina algunas de las k~SF's las cuales no producen SFM's, esto con el objetivo de reducir el número de SC's generadas. Para esto, Ahonen hace el crecimiento de cada una de las k~SF's hasta obtener una SFM. A partir del conjunto de las 1~SF's (palabras) y de una k~SF es posible generar una nueva *Secuencia Frecuente Candidata a Maximal* (SFCM); insertando cada una de las palabras al principio, en medio y al final de la k~SF. Una vez insertada una palabra se forma una nueva SFCM y su frecuencia tiene que ser verificada en la colección de documentos. La inserción de palabras en la SFCM continúa hasta que no crece más, con lo cual se ha encontrado

entonces una SFM. Si una k~SF es subsecuencia de alguna SFM no se hace el crecimiento de ese patrón, lo cual garantiza que no se va a encontrar esa misma SFM. Una vez terminada esta etapa se tiene asociada a cada k~SF una SFM. Después, utilizando el conjunto de SFM's encontradas se podan aquellas k~SF's que no van a generar nuevas SFM's por estar ya incluidas en algunas de las SFM's previamente encontradas, por lo que pueden ser eliminadas del conjunto de k~SF's, reduciendo la generación de SC's. Esto no garantiza que ya se encontraron todas las SFM's por lo que es necesario continuar con la generación de los $(k+1)$~SFM's hasta que no se generen más SC's. Sin embargo, la etapa de poda no es muy clara en su publicación [Ahonen 1999], la cual, después de un análisis en la publicación [Ahonen 1999b], se encontró que existían inconsistencias en el algoritmo MineMFS. Esta observación se deriva incluso del ejemplo presentado en su artículo [Ahonen 1999b], tabla 2.4, en el cual se buscan SFM's con GAP=2 y un umbral de frecuencia de 2; este ejemplo se muestra en la siguiente colección de documentos, donde las palabras son representadas por caracteres:

Tabla 2.4 Colección de documentos presentada por Ahonen en el artículo [Ahonen 1999b].

Documento 1=A,B,C,D,E
Documento 2=P,B,C,D,K
Documento 3=A,B,C,H,D,K
Documento 4=P,B,C,D,E,N
Documento 5=P,B,C,K,E,L,M
Documento 6=R,H,K,L,M

En [Ahonen 1999b], al finalizar la explicación del ejemplo, Ahonen menciona que todas las SFM's encontradas son:

2~SFM's	***3~SFM's***	***4~SFM's***	***4~SFM's***
$\langle H,K \rangle$	$\langle K,L,M \rangle$	$\langle A,B,C,D \rangle$	$\langle B,C,D,K \rangle$
		$\langle B,C,D,E \rangle$	$\langle P,B,C,D \rangle$

Sin embargo, si no es llevada a cabo la poda, lo cual incluye el crecimiento de una k~SF a un SFM, entonces el Algoritmo de Ahonen es igual que GSP, en este caso las SFM's encontradas serían:

2~*SFM's*	**3~*SFM's***	**3~*SFM's***	**4~*SFM's***
$\langle P,D\rangle$	$\langle A,B,D\rangle$	$\langle P,B,E\rangle$	$\langle A,B,C,D\rangle$
$\langle H,K\rangle$	$\langle A,C,D\rangle$	$\langle P,B,K\rangle$	$\langle B,C,D,E\rangle$
$\langle K,M\rangle$	$\langle B,D,E\rangle$	$\langle P,C,D\rangle$	$\langle B,C,D,K\rangle$
	$\langle B,D,K\rangle$	$\langle P,C,E\rangle$	$\langle P,B,C,D\rangle$
	$\langle P,B,D\rangle$	$\langle P,C,K\rangle$	$\langle P,B,C,E\rangle$
		$\langle K,L,M\rangle$	$\langle P,B,C,K\rangle$

Del ejemplo anterior se aprecia que la SFM $\langle P,D\rangle$ encontrada por GSP no fue encontrada por Ahonen, esto sucedió porque a partir de la SF=$\langle P,D\rangle$ se hizo el crecimiento hasta encontrar la SFM=$\langle P,B,C,D\rangle$, lo cual sugiere que la SF=$\langle P,D\rangle$ está contenida en $\langle P,B,C,D\rangle$, con lo cual se puede pensar que a partir de $\langle P,B,C,D\rangle$ se puede generar la secuencia frecuente $\langle P,D\rangle$. Cabe recordar que una de las características importantes de las SFM's es su representación compacta con respecto al conjunto de SF's, es decir, que a partir de ellas se puede generar todo el conjunto de SF's. Sin embargo, a partir de las SFM's encontradas por Ahonen no sería posible generar a todo el conjunto de SF's. Por ejemplo, la SFM=$\langle A,B,C,D\rangle$ generaría la SF=$\langle A,D\rangle$, la cual no existe en la colección de documentos, puesto que sólo se presenta en el documento 1, y en caso del documento 3 la restricción GAP=2 no permite obtener a la secuencia $\langle A,D\rangle$ como válida.

La inconsistencia presentada en el algoritmo de Ahonen se debe a que los algoritmos que NO contemplan la restricción GAP definen que una SF es *maximal* cuando no es subsecuencia de alguna otra SF, lo cual sólo es válido en SF's sin restricción GAP. Sin embargo, cuando los algoritmos SI contemplan la restricción GAP, entonces una SF es *maximal* cuando no es una *subsecuencia contigua (subsecuencia con GAP=0)* de otra SF. Cabe señalar que, de manera similar a GSP, ésta es la definición adoptada en esta tesis de una SFM con restricción GAP. De esta manera, las SFM's encontradas permiten generan todo el conjunto de SF's, por ejemplo a partir de la SFM=$\langle A,B,C,D\rangle$ se pueden generar las siguientes SF's={$\langle A,B,C,D\rangle$, $\langle A,B,C\rangle$, $\langle B,C,D\rangle$, $\langle A,B\rangle$, $\langle B,C\rangle$, $\langle C,D\rangle$, $\langle A\rangle$, $\langle B\rangle$, $\langle C\rangle$, $\langle D\rangle$}. Esta inconsistencia en el algoritmo de Ahonen hace que sólo sea comparable con GSP cuando no es aplicado el paso de crecimiento del SFCM y su poda respectiva, lo cual da como resultado que los algoritmos de Ahonen y GSP sean iguales para efectos de comparación. Este tipo de inconsistencia en los algoritmos de minería de patrones con restricción GAP también ha sido señalada en [Perego 2003][Orlando 2004].

2.2.3 Algoritmo PrefixSpan

Aunque PrefixSpan no permite considerar la restricción GAP, se presenta en esta sección el algoritmo PrefixSPan [Pei 2001, 2004] por ser una de las aportaciones importantes en la minería de patrones secuenciales, ya que PrefixSpan evita el paso de generación de secuencias candidatas mediante el crecimiento de SF's. PrefixSpan está basado en una filosofía ***divide y vencerás*** para lo cual realiza la construcción recursiva de patrones secuenciales mediante el uso de bases de datos proyectadas (α-BD). Una base de datos proyectada α es el conjunto de subsecuencias en la base de datos que son sufijos de las secuencias que contienen al prefijo α, en su correspondiente base de datos proyectada. Una vez construida la proyección de la base de datos, se enlistan los elementos frecuentes que contiene esa base de datos proyectada, agregando de esta manera un elemento al crecimiento del patrón secuencial. Además, para cada elemento frecuente se proyecta nuevamente la base de datos, esto se hace hasta que ya no se encuentran elementos frecuentes. De esta manera, el espacio de búsqueda es reducido en cada paso.

Por ejemplo, de la colección de documentos de la tabla 1.1 con umbral de frecuencia de 3 PrefixSpan encuentra las SFM's de la siguiente manera. Primero se enlistan ordenadamente los elementos frecuentes de la base de datos. En este caso si utilizáramos un orden alfabético se tendría:

⟨*A*⟩, ⟨*BAD*⟩, ⟨*BUSH*⟩, ⟨*IS*⟩, ⟨*PRESIDENT*⟩, ⟨*THE*⟩

A partir de esta lista, se generan recursivamente las bases de datos proyectadas para cada elemento de la lista (α-BD). Por ejemplo, para el primer elemento α=⟨A⟩, se tendría:

⟨*A*⟩-BD

1. *BAD PRESIDENT*
2. *BAD PRESIDENT*
3. *GOOD PRESIDENT*
4. *BAD PRESIDENT*
5. *REALLY BAD PRESIDENT*

Una vez construida la base de datos proyectada ⟨α⟩-BD se generan sus SF's agregando al inicio de cada SF el prefijo α, en el caso de ⟨*A*⟩-BD, las *2*~SF's (en crecimiento) encontradas serían: ⟨*A, BAD*⟩, ⟨*A, PRESIDENT*⟩. De manera similar, este proceso se desarrolla para cada

elemento frecuente enlistado. Para el caso de la primera SF=⟨*A, BAD*⟩ se proyectaría la base de datos ⟨*A, BAD*⟩-BD, la cual quedaría así:

⟨*A, BAD*⟩-BD

1. *PRESIDENT*
2. *PRESIDENT*
3. ----------
4. *PRESIDENT*
5. *PRESIDENT*

Finalmente de esta base de datos se produce la SF=⟨*A, BAD, PRESIDENT*⟩. El proceso recursivo para la SF ⟨*A, BAD, PRESIDENT*⟩ termina aquí puesto que ya no es posible generar más proyecciones, terminando así el crecimiento de la SF. En la figura 2.1 se muestra cómo PrefixSpan realiza la búsqueda en forma de árbol para la base de datos proyectada ⟨*THE*⟩-BD con el mismo umbral de frecuencia de 3.

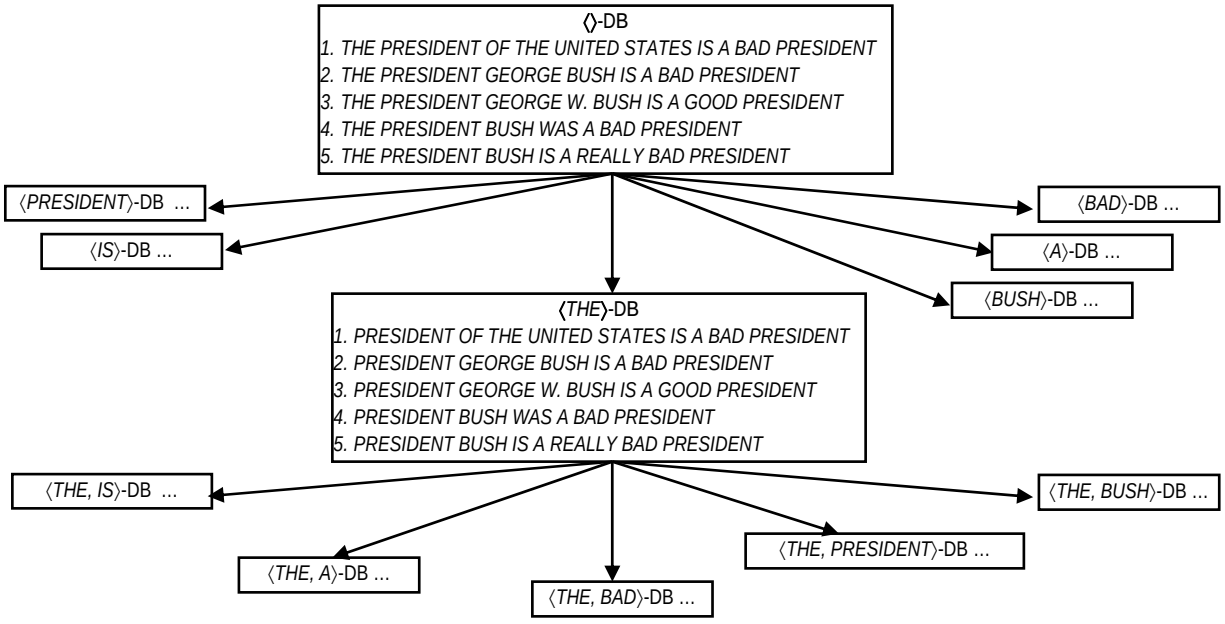


Figura 2.1 Árbol de bases de datos proyectada ⟨*THE*⟩-BD para PrefixSpan.

En el ejemplo anterior se aprecia cómo es dividido el problema de acuerdo al número de elementos frecuentes, reduciendo en cada proyección el espacio de búsqueda. Pero precisamente, el mayor costo computacional del algoritmo reside en el número de bases de datos proyectadas que se crean, que, en el peor caso, se harían tantas proyecciones como

patrones secuenciales encontrados. A pesar de esto, en los experimentos realizados en la publicación de PrefixSpan se supera el desempeño de GSP.

Cabe señalar que, a diferencia del problema perseguido en esta tesis, en PrefixSpan no es posible considerar la restricción GAP, además, de no generar solamente las SFM's lo cual implica un procesamiento adicional para seleccionar el conjunto de SFM's.

2.2.4 Algoritmo GenPrefixSpan

Claudia Antunes propone el algoritmo GenPrefixSpan [Antunes 2003][Antunes 2005] como una adaptación del algoritmo PrefixSpan [Pei 2004][Pei 2001] (uno de los mejores algoritmos dentro de la familia de los algoritmos de crecimiento de patrones) para que trabaje con la restricción GAP. La modificación de PrefixSpan consisten en incluir, en la base de datos proyectada α, todas las subsecuencias que contengan el prefijo α, a diferencia de PrefixSpan en donde sólo se incluye la primera ocurrencia del prefijo α. Una vez construida la base de datos proyectada se generan todas las SF's que permite el GAP.

Por ejemplo, considerando la colección de documentos de la tabla 1.1 con GAP de 1 y un umbral de frecuencia 3, en la figura 2.2 se presenta la base de datos proyectada para ⟨*THE*⟩-BD. En este caso GenPrefixSpan, a diferencia de PrefixSpan, toma en cuenta para ⟨*THE*⟩-BD los dos prefijos con ⟨*THE*⟩ del primer documento de ⟨⟩-BD debido a la restricción GAP. Una vez creada ⟨*THE*⟩-BD se enlistan los elementos frecuentes que cumplan con la restricción GAP, en este caso la SF={⟨*THE*, *PRESIDENT*⟩}. Luego se crearía ⟨*THE*, *PRESIDENT*⟩-DB la cual produciría la SF={⟨*THE*, *PRESIDENT*, *BUSH*⟩}. A su vez, ⟨*THE*, *PRESIDENT*, *BUSH*⟩-DB produciría la SF={⟨*THE*, *PRESIDENT*, *BUSH*, *A*⟩}. Con la ⟨*THE*, *PRESIDENT*, *BUSH*, *A*⟩-DB se genera la SF={⟨*THE*, *PRESIDENT*, *BUSH*, *A*, *BAD*⟩}. Con la ⟨*THE*, *PRESIDENT*, *BUSH*, *A*, *BAD*⟩-DB se genera la SF={⟨*THE*, *PRESIDENT*, *BUSH*, *A*, *BAD*, *PRESIDENT*⟩}. Finalmente uno de las SF's será ⟨*THE*, *PRESIDENT*, *BUSH*, *A*, *BAD*, *PRESIDENT*⟩.

Luego se proyectaría la base de datos ⟨*PRESIDENT*⟩-DB la cual sería idéntica a la proyección de ⟨*THE,PRESIDENT*⟩-DB mostrada en la figura 2.2. De este modo sabemos que al terminar el crecimiento la SF sería ⟨ *PRESIDENT*, *BUSH*, *A*, *BAD*, *PRESIDENT*⟩. De este ejemplo es posible observar cómo GenPrefixSpan tampoco determina directamente cuáles, de las SF's, son las maximales.

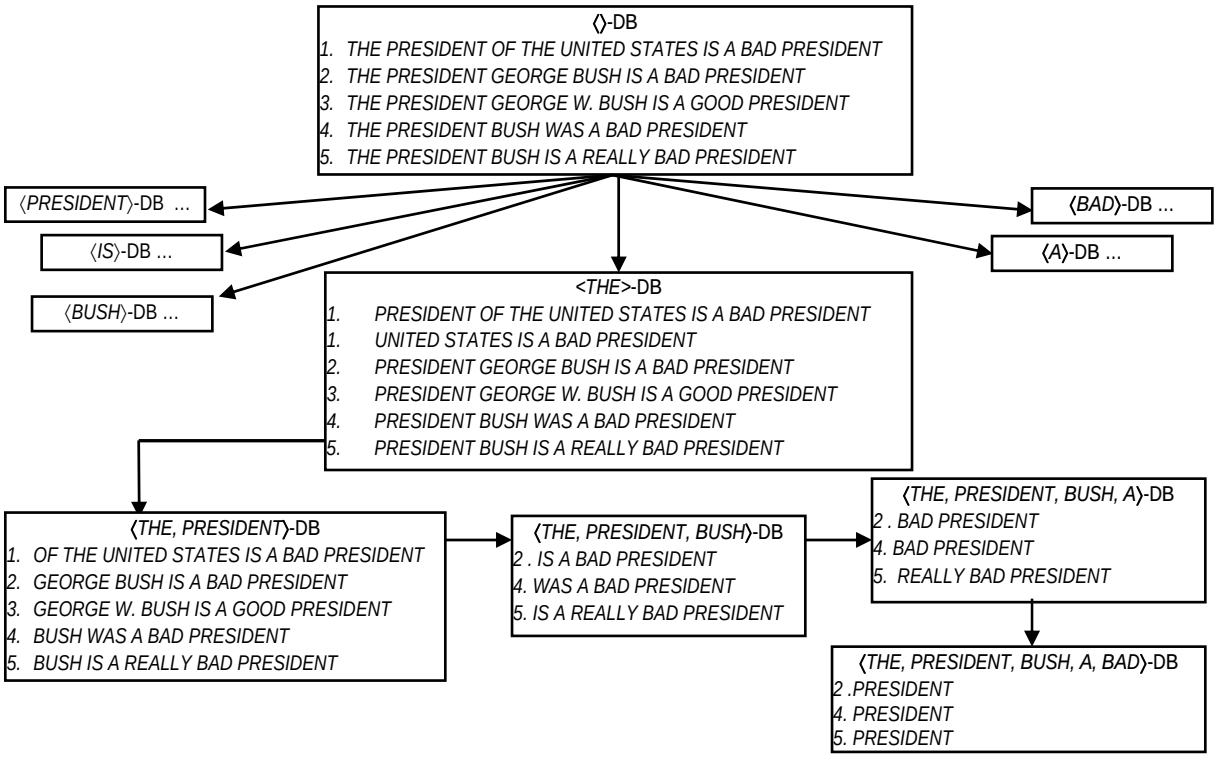


Figura 2.2 Árbol de las bases de datos proyectadas para ⟨*THE*⟩-BD de acuerdo a GenPrefixSpan.

Teniendo en cuenta que PrefixSpan está reportado como más rápido que GSP [Agrawal 1996], Antunes reporta [Antunes 2003] que GenPrefixSpan es más rápido que GSP cuando es aplicada la restricción GAP. Sin embargo, PrefixSpan obtiene su mejor rendimiento cuando la base de datos tiene un alfabeto pequeño debido al gran número de proyecciones que se generan y en el caso de secuencias de palabras (texto) se tendrían alfabetos de gran tamaño.

2.2.5 Algoritmo cSPADE

SPADE [Mohammed 2000] es un algoritmo de crecimiento de patrones, el cual está, según resultados reportados [Pei 2004], dentro de los que mejor desempeño muestra en cuanto a tiempo de ejecución se refiere; siendo más rápido que GSP, pero menos que PrefixSpan. La idea básica de SPADE consiste en registrar en una lista (*idlist*) las secuencias donde ocurre

cada elemento de la base de secuencias. Una vez encontradas 2~SF's, donde sus ocurrencias en las secuencias son guardadas en las listas ***idlist***, se pueden formar SC's de mayor longitud a partir de las listas ***idlist***, sin recorrer explícitamente la base de datos. Una vez encontradas dos k~SF's puede producir una SC si comparten sus $(k-1)$ sufijos. El problema con SPADE es el tamaño de las estructuras de datos generadas, lo cual es resuelto procesando por partes las estructuras de datos. Sin embargo, SPADE no permite considerar la restricción GAP, la cual es introducida en la versión cSPADE [Mohammed 2000a].

2.2.6 Árboles de Sufijos

Un árbol de sufijos es una estructura de datos que nos permite determinar eficientemente si una cadena es subcadena de otra [Arne 1999]. Un ***árbol de sufijos*** no es más que un árbol el cual guarda en sus ramas (caminos formados desde el nodo raíz hasta los nodos hoja) todos los sufijos (excepto el vacío) de una cadena, más el símbolo terminal '$'. Por ejemplo, a nivel de carácter la palabra *MISSISSIPPI* tiene como sufijos a:

1. MISSISSIPPI$
2. ISSISSIPPI$
3. SSISSIPPI$
4. SISSIPPI$
5. ISSIPPI$
6. SSIPPI$
7. SIPPI$
8. IPPI$
9. PPI$
10. PI$
11. I$

Una vez construido el árbol de sufijos (ver figura 2.3) se puede verificar de manera rápida si una cadena P es subcadena de *MISSISSIPPI*, para esto se verifica si la secuencia P existe como subrama del árbol, en caso contrario no es subsecuencia de *MISSISSIPPI*.

El objetivo de esta sección fue presentar a los árboles de sufijo debido a que, a primera vista, parecen resolver el problema de búsqueda de SFM's con GAP=0. Sin embargo, no es así ya que se requieren adaptaciones no triviales de esta estructura para resolver el problema que

se plantea en esta tesis, por lo cual esto constituye un tema de investigación, el cual no es el objetivo principal de la tesis.

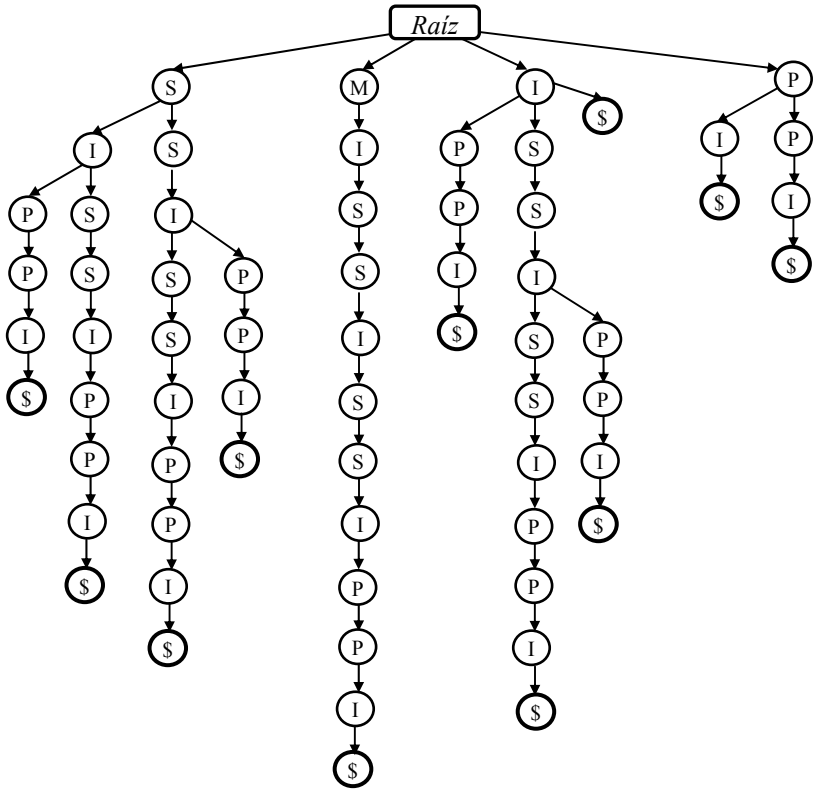


Figura 2.3 Árbol de sufijos para la palabra MISSISSIPPI. Los nodos hoja se muestran en línea más gruesa.

2.2.7 Ejemplos del uso de patrones secuenciales en texto

Uno de los primeros trabajos en cuanto a la aplicación de las SFM's con restricción GAP a texto, es el trabajo [Lent 1997] donde se descubren las tendencias en una base de datos de patentes en Estados Unidos, a este sistema se le conoce como PatentMiner. En este trabajo se aplicó inicialmente el algoritmo GSP para descubrir las tendencias de texto sobre un periodo de tiempo. A grandes rasgos el algoritmo consta de tres pasos: el primero es la identificación de las frases (secuencias) frecuentes usando minería de patrones secuenciales; en el segundo se genera un historial de estas frases; y en el tercero, se buscan las frases que satisfacen una tendencia específica, la cual es definida mediante un lenguaje llamado SDL.

En esta línea de trabajo se han encontrado en el 2005 tres trabajos, [Doucet 2004], [Doucet 2004a] y [Doucet 2004b]; sobre la utilidad de las SFM's en la recuperación de la información. Los dos primeros utilizan las SFM's en tareas de recuperación de información, y concluyen que para los experimentos que realizaron se obtuvo una mayor precisión en los documentos recuperados frente a los modelos que no toman en cuenta el orden de aparición de las palabras. En el tercer trabajo [Doucet 2004b], Antoine Doucet y Helena Ahonen estudian la utilidad de las SFM's en la recuperación de manera precisa de fragmentos de documentos XML usando EXTIRP, en donde uno de los módulos principales que intervienen en su método de recuperación es la extracción de las SFM's a partir del texto.

La tabla 2.5 muestra los algoritmos desarrollados para la minería de conjuntos y de secuencias frecuentes. En esta tabla es posible apreciar cómo evolucionaron algunos algoritmos, como es el caso de Apriori, AprioriAll, GSP, FreeSpan, PrefixSpan, GenPrefixSpan. También en esta tabla se muestra cuáles de los algoritmos desarrollados para la minería de patrones secuenciales permiten considerar la restricción GAP.

Algunas de las implementaciones de los algoritmos mostrados en la tabla 2.5 se pueden encontrar en [APRIORI][FIMI 2003, 2004][HIMALAYA].

También existen algoritmos de minería de patrones secuenciales que han implementado la restricción que los patrones secuenciales a descubrirse deben de cumplir con una expresión regular, como [Hunor 2003] [Lorincz 2003] [Minos 1999] [Pei 2002a], las cuales tendrían también gran sentido en tareas de procesamiento automático de texto. Sin embargo, los algoritmos de este tipo no permiten considerar la restricción GAP, a excepción de [Pei 2002a] que ya está considerado como el algoritmo GenPrefixSpan.

En la tabla 2.6 se resumen las características de los algoritmos presentados en esta sección. A partir de la tabla 2.6 es posible apreciar que solamente GSP, MineMFS(Ahonen) y GenPrefixSpan permiten considerar la restricción GAP. También es posible observar que el tamaño de las secuencias que se utilizaron para sus experimentos no es muy grande, entre los de mayor tamaño están las utilizadas por MineMFS y PrefixSpan. De igual manera, en estos dos algoritmos el tamaño de las SFM's encontradas en sus experimentos no es mayor de 20. Otro aspecto a considerar es el tamaño del alfabeto, en los experimentos de MineMFS se utilizó una colección con 50,000 elementos diferentes. Sin embargo, en los otros algoritmos se utilizaron colecciones con un alfabeto menor a 10,000 elementos. La tabla 2.6 muestra que la mayoría de los algoritmos consideran, en los experimentos presentados, tamaños pequeños de secuencias de la base de datos, de las SFM's y del alfabeto, por lo que es posible que no

obtengan un buen desempeño cuando sean aplicados a conjuntos de datos más grandes. Cabe mencionar que los tamaños de las secuencias, SFM's y del alfabeto, presentados en la tabla 2.6, son los tamaños presentados en los experimentos de cada artículo, lo que no quiere decir que son tamaños necesariamente restrictivos. Excepto por el algoritmo de Ahonen, el tamaño de las secuencias en la base de datos, de las SFM's y del alfabeto se pueden controlar porque existe un generador sintético [QUEST] de bases de secuencias que lo permite.

Tabla 2.5 Algoritmos para minería de conjuntos y secuencias frecuentes

Tipo de algoritmo	**Algoritmo**	**Minería de:**		
		Conjuntos frecuentes	**Patrones secuenciales sin Gap**	**Patrones secuenciales con Gap**
A priori o típicos	A priori [Agrawal 1994a]	✓		
	MSApriori *(Multiples Soportes)*	✓		
	AprioriAll [Agrawal 1994] (sucesor de Aprori)		✓	
	AprioriSome [Agrawal 1994]		✓	
	AprioriDynamicSome [Agrawal 1994]		✓	
	MAFIA[Burdick 2001]	✓		
	SPAM [Jay 2002]		✓	
	GSP [Agrawal 1996] (sucesor de AprioriAll)			✓
	MineMFS (Ahonen) [Ahonen 1999, 2005]			✓
	Pincel-Search [Dao 1998]	✓		
Crecimiento de patrones	SLPMiner [Seno 2002]		✓	
	SPADE [Mohammed 2000]		✓	
	FreeSpan [Han 2000]		✓	
	PrefixSpan sin Gap [Pei 2001, 2004] (sucesor de FreeSpan)		✓	
	GenPrefixSpan (PrefixSpan con Gap) [Antunes 2003, 2005]			✓
	cSPADE [Mohammed 2000a]			✓
	DELISP [Ming 2002, 2003]			✓
	MEMISP [Ming 2002a, 2003]		✓	

Tabla 2.6 Características de los algoritmos de minería de patrones secuenciales

Estrategia de búsqueda	Algoritmo	Secuencias formadas por	Restricción GAP	Tamaño de las secuencias en la BD	Tamaño de las SFM's	Tamaño del alfabeto
Típicos	**GSP**	Transacciones	Si	[10,20]	[4,8]	10,000
	Ahonen	Texto	Si	135	[2,22]	**50,000**
Crecimiento de patrones	**SPADE**	Transacciones	No	[10,20]	[4,8]	[500,1000]
	FreeSpan	Transacciones	No	[10,20]	4	10,000
	PrefixSpan sin GAP	Transacciones	No	[8,100]	4	[1000,10000]
	GenPrefixSpan	Transacciones	Si	10	4	[5,500]

2.3. Problema de investigación

La minería de patrones secuenciales tiene como objetivo la extracción de todas las secuenciales frecuentes de un conjunto de secuencias. El problema fundamental a atacar en esta disertación es el desarrollo de algoritmos para la búsqueda de secuencias frecuentes maximales en texto. La mayoría de los algoritmos de minería de patrones secuenciales han sido desarrollados para bases de datos de transacciones, a diferencia de nuestro caso, en el cual se plantea para documentos de texto. Este problema es muy importante debido a la gran cantidad de información almacenada en documentos. En nuestro caso nos interesan las secuencias frecuentes, pero que sean maximales, debido a que son una representación compacta del conjunto de todas las secuencias frecuentes.

El problema de minería de texto está definido sobre un conjunto de secuencias, sin embargo al considerar un documento como una secuencia de palabras, entonces existiría un número importante de SFM's que podrían ser extraídas a partir de un sólo documento. Esto último nos da otra característica importante al problema atacado, por lo que también habrá que desarrollar algoritmos para la búsqueda de SFM's en un sólo documento de texto.

Otra característica importante a considerar es que para nuestro problema es necesario buscar SFM's con restricción GAP pequeña, porque SFM's con GAP grande difícilmente pueden ayudar al análisis de texto. Por ejemplo, podríamos tener una SFM en donde su primera palabra esté en la primera página y la siguiente palabra corresponda a 20 páginas más adelante; perdiendo de esta manera el contexto y el sentido entre las palabras de la SFM. Además de que el número de secuencias frecuentes descubiertas aumentaría, lo que

implica también un aumento en el tiempo de extracción. Como se aprecia en la tabla 2.5, son pocos los algoritmos de minería de patrones secuenciales que contemplan esta restricción.

La mayoría de los algoritmos en minería de patrones secuenciales son aplicados a problemas con alfabetos pequeños. En el caso de texto, el tamaño del alfabeto puede ser más grande. Por ejemplo, la colección Reuters-21578 que consiste de 21,578 noticias tiene alrededor de 50,000 palabras, y en los experimentos reportados en [Agrawal 1994][Pei 2001][Antunes 2003] el tamaño del alfabeto está entre 500 y 1000 ítems.

En el caso de bases de datos de transacciones las secuencias no son muy largas, a diferencia del texto donde pueden ser muy largas. En los experimentos reportados en [Agrawal 1996][Pei 2001][Mohammed 2000][Antunes 2003] trabajan con secuencias de un tamaño promedio entre 10 y 100 ítems. En nuestro caso un documento con 100 palabras no es muy grande. Este aspecto tiene que ser tomado en cuenta en el desarrollo de algoritmos para buscar secuencias frecuentes maximales de texto. Es decir, algunos de los aspectos importantes a tener en cuenta en el desarrollo de estos algoritmos son:

- Tiempo de procesamiento
- Espacio requerido
- Tamaño del alfabeto
- Restricción Gap
- Tamaño del texto y de la colección de textos
- Tamaño de las SFM's encontradas en los textos

Todo esto planteó la pregunta de esta investigación:

> ¿Es posible encontrar nuevas propiedades, estructuras de datos y estrategias de búsqueda que permitan desarrollar algoritmos más eficientes que los actualmente disponibles, los cuales resuelvan el problema de minería de patrones secuenciales maximales para el análisis de documentos?

Por "*más eficiente*" se entiende que los algoritmos a desarrollarse empleen menos tiempo y, posiblemente, menos espacio que los actualmente desarrollados.

2.4. Objetivos

El objetivo principal de este trabajo consistió en desarrollar algoritmos que permitan obtener los patrones secuenciales maximales con restricción GAP tanto en un documento como en una colección de documentos de manera más rápida que los actuales.

2.5. Sumario

Con el propósito de introducir a la minería de patrones secuenciales se mencionó la minería de patrones que es realizada sobre conjuntos, mostrando ejemplo de ambos tipos de minería. Después se expuso el estado del arte de los algoritmos que han sido desarrollados tanto para conjuntos como para secuencias, el cual puede resumirse en la tabla 2.5. De la tabla 2.5 se muestra que son pocos los algoritmos que contemplan la búsqueda de SFM's con restricción GAP. De esta manera, en la tabla 2.4 se muestran las características de los algoritmos de patrones secuenciales, donde es posible observar los tipos y tamaños de datos que presentan en su experimentación. Finalmente se establece el problema, la pregunta de investigación y el objetivo principal de esta investigación.

CAPÍTULO 3

Descubrimiento de SFM's en una colección de documentos

En este capítulo se presentan los algoritmos desarrollados en esta tesis para el descubrimiento de secuencias frecuentes maximales (SFM's) a partir de una colección de documentos con restricción GAP=0 (Dimasp-C_0) y con restricción GAP=n (Dimasp-C_n). En ambos casos, primero se plantea el problema, luego se describe el nuevo algoritmo propuesto y se compara experimentalmente contra otros algoritmos de búsqueda de SFM's.

3.1. Descubrimiento de SFM's en una colección de documentos para GAP=0

En esta sección se presenta Dimasp-C_0 (***Di**scovering **Ma**ximal **S**equential **P**atterns in a Document **C**ollection for GAP=**0***) que es un algoritmo para el descubrimiento de todas las

secuencias frecuentes maximales con GAP=0 en una base de datos de documentos (BDD). En la siguiente sección se plantea formalmente el problema. Posteriormente, se introduce Dimasp-C_0 y se compara contra otros algoritmos que permiten encontrar las SFM's con GAP=0 en una BDD.

3.1.1. Definición del problema

Una ***secuencia*** S, denotada por $\langle s_1,s_2,...,s_k\rangle$, es una lista ordenada de k elementos. Una secuencia de ***longitud*** k es denominada k~secuencia.

Sean $P=\langle p_1,p_2,\ldots,p_t\rangle$ y $S=\langle s_1,s_2,\ldots,s_m\rangle$ secuencias, P es una ***subsecuencia con GAP=0*** de S, denotado como $P\subseteq_0 S$ si existe un entero $i \geq 1$ tal que $p_1=s_i, p_2=s_{i+1}, p_3=s_{i+2}, \ldots, p_t=s_{i+(t-1)}$.

Un documento W se puede considerar como una secuencia de palabras, denotado también como $\langle w_1,w_2,\ldots,w_t\rangle$.

La ***frecuencia*** de una secuencia S en una colección de documentos $\{W_1, W_2, \ldots, W_j\}$ considerados como secuencias, denotada por S_f o $\langle s_1,s_2,...,s_t\rangle_f$, es el número de documentos en los cuales S aparece por lo menos una vez, esto es, $S_f=|\{ W_i \mid S\subseteq_0 W_i \}|$.

Dado un umbral definido por el usuario (β), una secuencia S es ***frecuente*** si $S_f \geq \beta$. Una secuencia frecuente S es ***maximal*** si S no es subsecuencia de alguna otra secuencia frecuente. A una secuencia frecuente maximal (SFM) también se le conoce como ***patrón secuencial maximal***.

Dada una colección de documentos, el problema abordado en esta sección es el descubrimiento de todas las secuencias frecuentes maximales con GAP=0 para un umbral β dado.

3.1.2. Algoritmo propuesto (Dimasp-C_0).

El algoritmo Dimasp-C_0 es el resultado de buscar una estructura que permitiera encontrar de manera rápida las secuencias frecuentes de menor longitud y que también permitiera encontrar las SFM's creciendo las secuencias frecuentes de menor longitud. Construir una estructura con estas características evitaría generar secuencias candidatas como en GSP

[Agrawal 1996]. En algunos casos, por ejemplo [Pei 2001][Antunes 2003], las estructuras de datos utilizadas dependen del umbral de frecuencia, por lo que todo el trabajo realizado es desechado una vez que termina la búsqueda para ese umbral. Por esto, sería deseable tener una estructura independiente del umbral de frecuencia. Aun más, sería conveniente que la construcción de la estructura sea lo más simple posible, pues esto tendría beneficios tanto en el tiempo como en el espacio requerido para su construcción. Una de las características más importantes que debería permitir la estructura es identificar fácilmente las secuencias frecuentes, de las que no lo son, permitiendo ir directamente a la búsqueda de SFM's.

Para la construcción de la estructura de datos Dimasp-C_0 representa cada documento mediante los pares de palabras que contiene. Por ejemplo, el documento D_3=$\langle a,b,c,d\rangle$ se puede representar por sus pares de palabras $\langle a,b\rangle$, $\langle b,c\rangle$ y $\langle c,d\rangle$. En la tabla 3.1 se muestra la representación de documentos mediante pares de palabras para una colección de 6 documentos.

Tabla 3.1 Ejemplo de una colección de documentos y su representación mediante pares de palabras. El subíndice en los pares de palabras representa la frecuencia de ese par en la colección de documentos.

Documento	Documento original	Representación del documento mediante sus pares de palabras contiguos
D_1	$\langle a,b,f\rangle$	$\langle a,b\rangle_4\ \langle b,f\rangle_2$
D_2	$\langle b,c,d\rangle$	$\langle b,c\rangle_5\ \langle c,d\rangle_3$
D_3	$\langle a,b,c,d\rangle$	$\langle a,b\rangle_4\ \langle b,c\rangle_5\ \langle c,d\rangle_3$
D_4	$\langle a,b,c,d,e,a,b,f\rangle$	$\langle a,b\rangle_4\ \langle b,c\rangle_5\ \langle c,d\rangle_3\ \langle d,e\rangle_1\ \langle e,a\rangle_1\ \langle a,b\rangle_4\ \langle b,f\rangle_2$
D_5	$\langle b,c,e\rangle$	$\langle b,c\rangle_5\ \langle c,e\rangle_2$
D_6	$\langle a,b,c,e\rangle$	$\langle a,b\rangle_4\ \langle b,c\rangle_5\ \langle c,e\rangle_2$

Si el umbral de frecuencia β es 2 entonces, en la colección de documentos de la tabla 3.1, $\langle a,b,c,d\rangle$ es una *SFM* la cual también puede ser representada mediante los pares de palabras $\langle a,b\rangle_4$, $\langle b,c\rangle_5$ y $\langle c,d\rangle_3$, los cuales tienen cada uno una frecuencia mayor o igual a β. Esto nos permite saber que es posible localizar la *SFM* si encontramos primero todas las ocurrencias de la secuencia $\langle a,b\rangle$ en toda la colección de documentos, donde el número de ocurrencias tiene que ser mayor o igual a β. Por ejemplo, en la figura 3.1 están señaladas las ocurrencias del par de palabras $\langle a,b\rangle$ de la colección de documentos de la tabla 3.1.

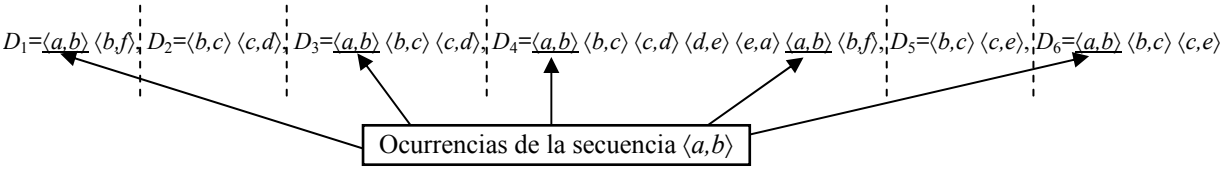


Figura 3.1 Ocurrencias de la secuencia ⟨*a,b*⟩ en la colección de documentos de la tabla 3.1.

Una vez encontrado el conjunto de todas las ocurrencias de la secuencia ⟨*a,b*⟩, para encontrar la SFM ⟨*a,b,c,d*⟩ habrá que descartar aquellas ocurrencias que no continúen con el par de palabras ⟨*b,c*⟩, de esta manera el nuevo conjunto de ocurrencias corresponde a la secuencia ⟨*a,b,c*⟩, lo cual se muestra en la figura 3.2. En la figura 3.2 puede verse mediante los pares de palabras subrayados la forma en que crece la SF.

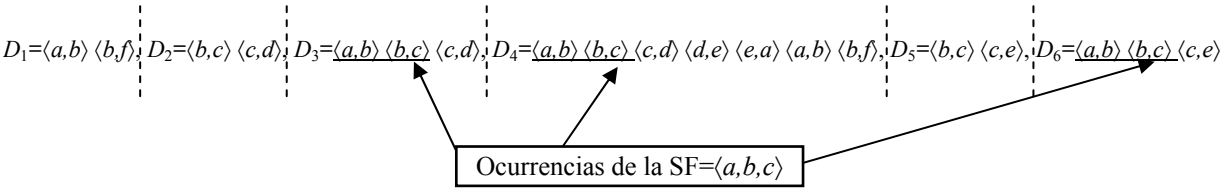


Figura 3.2 Ocurrencias de la secuencia ⟨*a,b,c*⟩ en la colección de documentos de la tabla 3.1.

De manera similar, una vez encontrado el conjunto de ocurrencias de la secuencia ⟨*a,b,c*⟩, habrá que descartar aquellas ocurrencias que no continúen con el par de palabras ⟨*c,d*⟩, de esta manera el nuevo conjunto de ocurrencias permite encontrar la SFM ⟨*a,b,c,d*⟩ en la colección de documentos, lo cual se muestra en la figura 3.3.

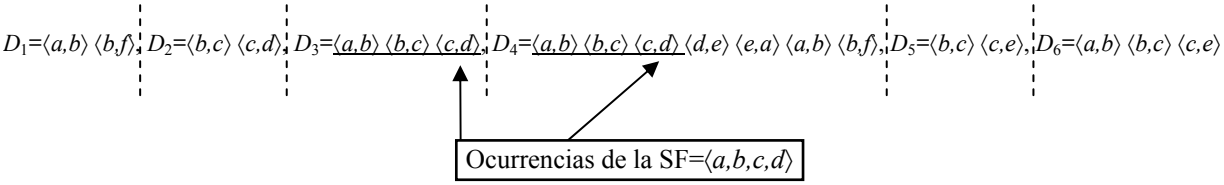


Figura 3.3 Ocurrencias de la secuencia ⟨*a,b,c,d*⟩ en la colección de documentos de la tabla 3.1.

Siguiendo esta idea, Dimasp-C_0 construye una estructura de datos a partir de los documentos de la BDD. Para ello, se utiliza una lista (llamada *Lista* Δ) la cual liga todas las ocurrencias de la secuencia ⟨*a,b*⟩ en toda la BDD, de manera que a partir de *Lista* Δ sea posible recuperar de manera rápida las posiciones donde ocurre la secuencia ⟨*a,b*⟩. Así, la frecuencia de la secuencia ⟨*a,b*⟩ estará determinada por el número de documentos diferentes que hay en *Lista* Δ. La figura 3.4 muestra una representación de la forma en que *Lista* Δ liga las ocurrencias de la secuencia ⟨*a,b*⟩ en la BDD, en este caso *Lista* Δ tiene una frecuencia de 4.

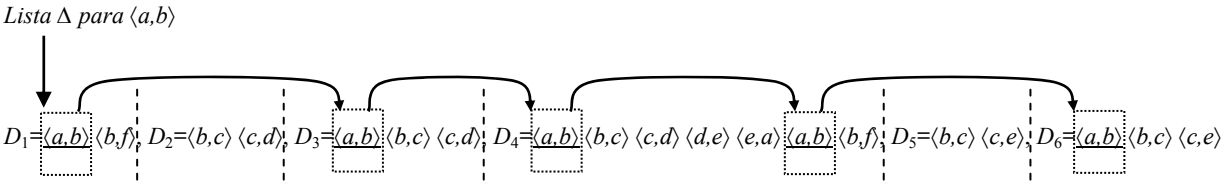


Figura 3.4 Representación de *Lista* Δ para el par de palabras ⟨*a,b*⟩ en la colección de documentos de la tabla 3.1.

Nótese que *Lista* Δ liga todas las ocurrencias del par ⟨*a,b*⟩ incluyendo todas las ocurrencias dentro de cada documento, como sucede con D_4. A partir de *Lista* Δ es posible crecer la secuencia ⟨*a,b*⟩ a la secuencia ⟨*a,b,c*⟩, siempre y cuando los pares de palabras ⟨*a,b*⟩ y ⟨*b,c*⟩ se presenten en este orden al menos β veces en la BDD. Para el ejemplo anterior, *Lista* Δ cambiaría quedando como se muestra en la figura 3.5.

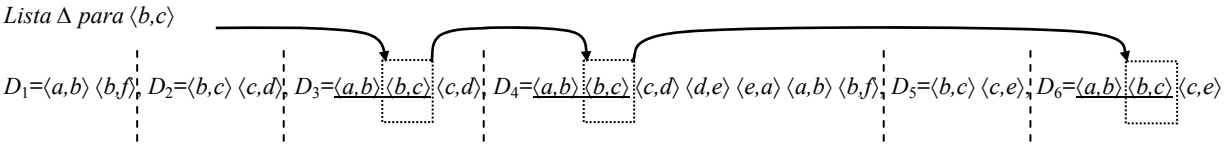


Figura 3.5 *Lista* Δ para el par de palabras ⟨*b,c*⟩ en la colección de documentos de la tabla 3.1.

Utilizando la *Lista* Δ actual es posible realizar el crecimiento de la secuencia ⟨*a,b,c*⟩ a ⟨*a,b,c,d*⟩, quedando la *Lista* Δ como se muestra en la figura 3.6. Finalmente, la secuencia ⟨*a,b,c,d*⟩ ya no puede crecer más, pues es una SFM.

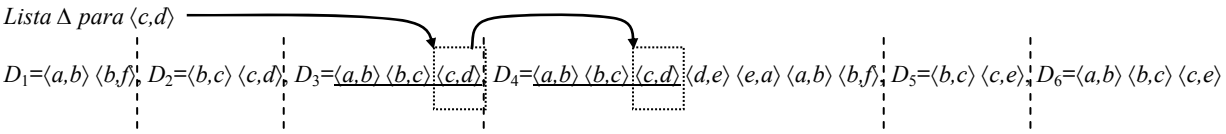


Figura 3.6 *Lista* Δ para el par de palabras ⟨*a,b,c*⟩ en la colección de documentos de la tabla 3.1.

En general, Dimasp-C_0 mantiene una *Lista* Δ para cada par de palabras diferentes de la BDD, las cuales son almacenadas en un arreglo. En este arreglo es posible acceder a las características de cada *Lista* Δ, esto es, el par de palabras, su frecuencia y la lista de ocurrencias en toda la BDD. Utilizando el ejemplo anterior, se muestra en la figura 3.7 una representación del arreglo de las *Listas* Δ junto con sus características, para cada par de palabras de la BDD.

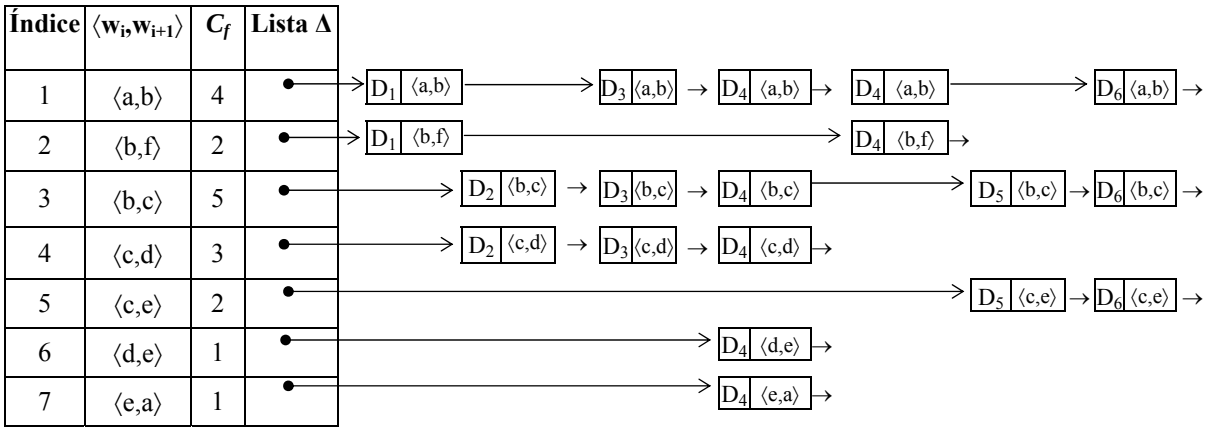


Figura 3.7 Arreglo de *Listas* Δ para los pares de palabras deferentes de la colección de documentos utilizada en la tabla 3.1.

Hasta este momento, se observa que la estructura de datos de Dimasp-C_0 está basada en representar los documentos mediante los pares de palabras que contienen. Sin embargo, los pares no preservan su estructura secuencial, por lo que todavía no es posible realizar el crecimiento de la secuencia frecuente. Esto puede resolverse utilizando una liga entre los pares de palabras de cada documento, lo que permite mantener el orden secuencial. La figura 3.8 nos muestra de manera simplificada la representación adoptada por Dimasp-C_0 para la colección de documentos de la tabla 3.1. Esta estructura preserva el orden secuencial de los pares de palabras que contiene.

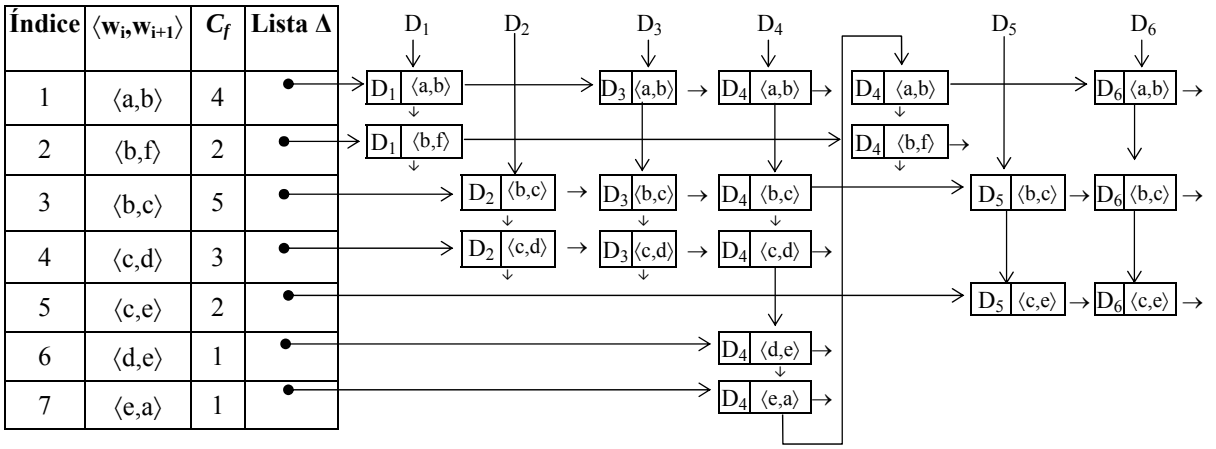


Figura 3.8 Estructura de datos construida en Dimasp-C_0 en la cual están ligados los pares de palabras contiguos, para la colección de documentos de la tabla 3.1.

Con el propósito de reducir la información almacenada en *Lista* Δ, es posible sustituir los pares de palabras por el índice donde se encuentra el par en el arreglo. También es posible sustituir el identificador del documento por el número de documento analizado, lo cual se muestra en la figura 3.9. De esta manera, a partir de la estructura de la figura 3.9 es posible hacer el descubrimiento de las SFM's, pues es posible recuperar todas las ocurrencias de un par de palabras en la BDD, sin perder el orden secuencial de los documentos.

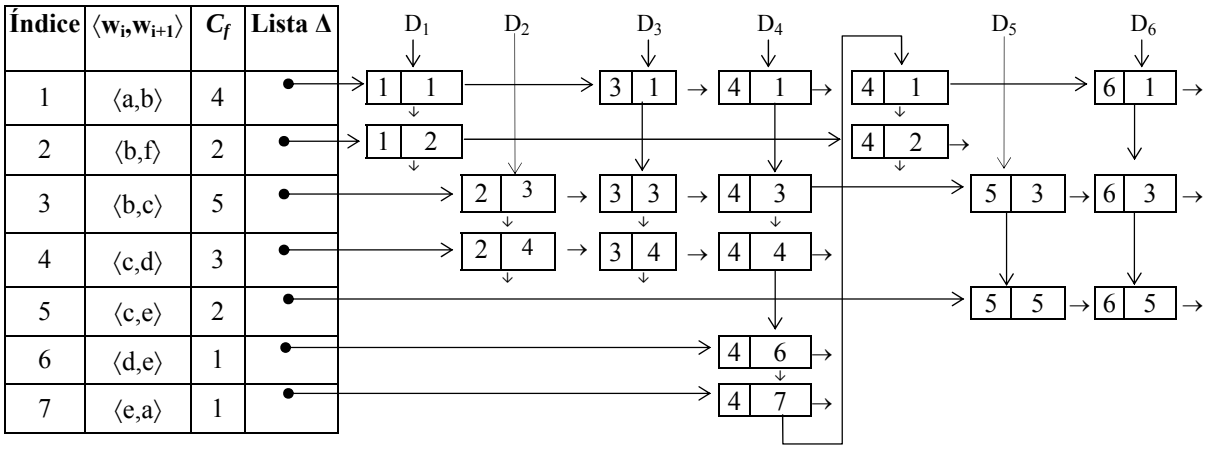


Figura 3.9 Estructura de datos construida en Dimasp-C_0 en la cual se sustituyeron los pares de palabras por su índice en el arreglo y los identificadores de los documentos por su número.

Como se mostró, la idea básica de Dimasp-C_0 consiste en encontrar todas las SFM's en una estructura de datos —construida a partir de la base de datos de documentos (BDD)— la cual guarda todos los pares distintos de palabras contiguas que aparecen en los documentos, sin perder el orden secuencial. Dado un umbral β, especificado por el usuario, Dimasp-C_0 revisa si un par de palabras es frecuente, en cuyo caso es una secuencia frecuente (SF) de longitud 2 (2~SF). A partir de cada SF, Dimasp-C_0 crece, lo más posible, cada secuencia con el objetivo de determinar todas las SF's que contienen al par de palabras como prefijo. Una vez que una SF no puede crecer más, sólo falta determinar si esa SF no es subsecuencia de alguna otra SF, en cuyo caso es una SFM.

Dimasp-C_0 está dividido en 4 etapas. En la primera etapa, a partir de la BDD se transforman y preparan los datos asignando identificadores numéricos a las palabras y a los pares de palabras para que la segunda etapa trabaje de manera más eficiente. En la segunda etapa, a partir de los pares de palabras de la BDD se construye la estructura de datos que permitirá buscar las SFM's. En la tercera etapa, de acuerdo al umbral β especificado por el usuario, para cada par de palabras de la BDD se busca la SF más larga que se pueda formar a partir

de él, pues entre este conjunto de SF's largas, se encuentran las SF's maximales. Por último, en la cuarta etapa se seleccionan, a partir del conjunto de SF's encontradas, las que son maximales. Cabe señalar, que las palabras que son frecuentes y que no están contenidas en alguna SFM's entonces esas palabras son SFM's, las cuales son determinadas después de la tercera etapa.

3.1.2.1 Primera etapa, transformación de los documentos

Con el objetivo de ahorrar espacio y facilitar el manejo de palabras, en la primera etapa se realiza la substitución de cada palabra diferente de la BDD por un identificador numérico. Para esto se utiliza un ***arreglo*** donde se guardan las palabras diferentes y sus frecuencias en la colección de documentos. La frecuencia de una palabra corresponde al número de documentos diferentes donde ésta aparece. Así, la posición de una celda en el arreglo representa el identificador numérico utilizado en lugar de esa palabra. Por ejemplo, en la tabla 3.2 se tienen cuatro documentos de texto junto con los identificadores asignados para cada palabra diferente de la colección.

Tabla 3.2 Ejemplo de una base de documentos y de su representación con identificadores numéricos.

D_i	**Base de documentos (BDD)**	**Identificadores numéricos**
1	*FROM GEORGE WASHINGTON TO GEORGE W. BUSH ARE 43 PRESIDENTS*	⟨1,2,3,4,2,5,6,7,8,9⟩
2	*WASHINGTON IS THE CAPITAL OF THE UNITED STATES*	⟨3,10,11,12,13,11,14,15⟩
3	*GEORGE WASHINGTON WAS THE FIRST PRESIDENT OF THE UNITED STATES*	⟨2,3,16,11,17,18,13,11,14,15⟩

Así como a cada palabra w_i de la BDD se le asigna un identificador numérico, también en esta etapa a cada par de palabras $\langle w_{i-1},w_i\rangle$ de la BDD se le asigna un identificador numérico.

Para la asignación de identificadores en lugar de palabras se emplea una estructura de datos de tipo árbol que, además de permitir obtener el identificador de cada palabra w_i, también permite obtener el índice (en el arreglo de pares de palabras) correspondientes a $\langle w_{i-1},w_i\rangle$. El árbol tiene en cada una de sus ramas a cada una de las palabras w_i (más el símbolo terminal '$') de la BDD. De esta manera, la palabra formada por una rama es sustituida por el ***identificador numérico*** del nodo hoja de esa rama. En la figura 3.10 se muestra el árbol construido para asignar identificadores a las palabras de la colección de la tabla 3.2.

Para la asignación de identificadores numéricos del par de palabras $\langle w_{i-1},w_i\rangle$ de la BDD se puede utilizar el mismo el árbol de palabras descrito anteriormente. En este caso, el árbol

mantiene en cada rama al par de palabras "$w_{i-1}\$w_i\$$". Una vez agregada la palabra "$w_{i-1}\$w_i\$$" como rama del árbol se puede conocer al identificador numérico del nodo hoja.

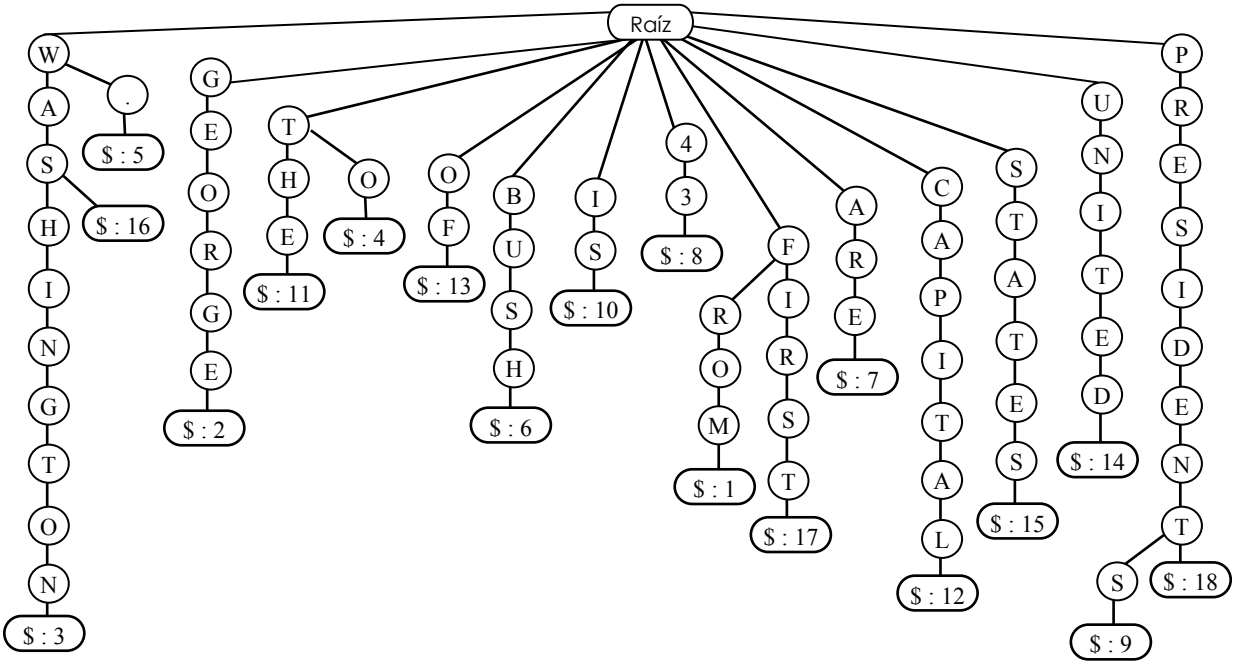


Figura 3.10 Árbol construido en la primera etapa de Dimasp-C_0. El árbol está construido para las palabras de la colección de documentos de la tabla 3.2. El óvalo indica los ídentificadores numéricos de las palabras. El símbolo terminal usado es '$'.

Si n es el número de palabras en la colección de documentos, entonces el peor caso en esta etapa se presenta cuando existen n palabras diferentes, lo cual forma también n–1 pares de palabras diferentes. Como se explicó anteriormente, cada palabra "$w_i\$$" y cada par de palabras "$w_{i-1}\$w_i\$$" de la BDD se agrega al árbol en forma de rama. En este caso, el árbol tendría a lo más n ramas, por lo que esta etapa presenta en el peor caso un espacio de orden O(n). Como se puede ver, cada palabra "w_i" de la BDD es procesada dos veces (una vez para obtener el identificador de "$w_i\$$" y otra vez para obtener el identificador de "$w_{i-1}\$w_i\$$") y cada vez se genera a lo más una rama, por lo que esta etapa presenta en el peor caso un tiempo O(n).

3.1.2.2 Segunda etapa, construcción de la estructura de datos

En la segunda etapa, Dimasp-C_0 construye una estructura de datos a partir de la BDD (como se mostró en la sección 3.1), almacenando todos los pares contiguos de palabras $\langle w_i, w_{i+1}\rangle$ que

aparecen en los documentos, además de almacenar información adicional para preservar el orden secuencial. La estructura de datos está compuesta de un ***arreglo*** de *Listas Δ*, en donde se tiene para cada celda el par de palabras $C=\langle w_i,w_{i+1}\rangle$, la ***frecuencia*** del par (C_f), una ***marca*** Booleana (utilizada en la tercera etapa) y una *Lista Δ* de nodos $\delta(\langle w_i,w_{i+1}\rangle)$. Un ***nodo*** δ (ver figura 3.11) representa la ocurrencia de un par de palabras $\langle w_i,w_{i+1}\rangle$ en la BDD y almacena el identificador del documento ($\delta.Id$), el ***índice*** ($\delta.index$) de la celda donde el par aparece en el arreglo, una liga ($\delta.SamePair$) para mantener la *Lista Δ* y una liga ($\delta.NextPair$) para preservar el orden secuencial de los pares con respecto al documento donde aparecen. Por lo tanto, el número de documentos diferentes en *Lista Δ* será la frecuencia del par C_f. El algoritmo de la segunda etapa se muestra en la figura 3.12. La figura 3.13 muestra la estructura de datos construida en la segunda etapa usando la base de datos de la tabla 3.2.

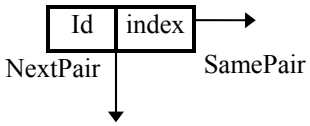


Figura 3.11 Estructura de un nodo δ

Etapa 2: Algoritmo para construir la estructura de datos a partir de BDD
Entrada: Una base de datos de documentos (BDD) transformada en la etapa 1
Salida: La estructura de datos de la BDD
Para *todos los documentos* $D_j \in BDD$ **hacer**
    Arreglo ← Agregar el documento (D_j) al arreglo
fin-para
Etapa 2.1: Algoritmo para agregar un documento a la estructura de datos
Entrada: Un documento D_j; BDD transformada; **Salida:** Estructura de datos
Para *todos los pares de palabras* $\langle w_i,w_{i+1}\rangle \in D_j$ **hacer**
    $\delta_i(\langle w_i,w_{i+1}\rangle)$ ← *Crear el nuevo nodo* δ *correspondiente a* $\langle w_i, w_{i+1}\rangle$
    $\delta_i(\langle w_i,w_{i+1}\rangle).Id \leftarrow j$ //Asignar el identificador del documento al nodo
    índice ← $Arreglo[\langle w_i,w_{i+1}\rangle]$ //Obtener el índice de la celda de $\langle w_i,w_{i+1}\rangle$, calculado en la etapa 1
    $\delta_i(\langle w_i,w_{i+1}\rangle).index \leftarrow$ *índice* //Asignar el índice al nodo δ
    $\alpha \leftarrow Arreglo[índice].Lista\ \Delta$ //Obtener el primer nodo de la lista Δ
    Si $\delta_i.Id \neq \alpha.Id$ **entonces** *identificador del documento es nuevo en Lista Δ*
        $Arreglo[índice].Lista\ \Delta.C_f$ ++ //incrementar la frecuencia de la celda
    $Arreglo[índice].Lista\ \Delta \leftarrow \delta_i.SamePair \leftarrow \alpha$ //Ligar el nodo al comienzo de Lista Δ
    Establecer la asociación $\delta_{i-1}(\langle w_{i-1}, w_i\rangle) \rightarrow \delta_i(\langle w_i, w_{i+1}\rangle)$, *es decir,* $\delta_{i-1}.NextPair \leftarrow \delta_i$
fin-para

Figura 3.12 Algoritmos de la segunda etapa donde es construida la estructura de datos para Dimasp-C_0

ndice	⟨w_i,w_{i+1}⟩	C_f	¿El par fue usado?	Lista Δ
1	⟨From,George⟩	1	No	
2	⟨George,Washington⟩	2	No	
3	⟨Washington,to⟩	1	No	
4	⟨to,George⟩	1	No	
5	⟨Washington,is⟩	1	No	
6	⟨is,the⟩	1	No	
7	⟨the,capital⟩	1	No	
8	⟨capital,of⟩	1	No	
9	⟨Washington,was⟩	1	No	
10	⟨was,the⟩	1	No	
11	⟨the,first⟩	1	No	
12	⟨first,President⟩	1	No	
14	⟨President,of⟩	1	No	
15	⟨of,the⟩	2	No	
16	⟨the,United⟩	2	No	
17	⟨United,States⟩	2	No	
20	⟨George,W.⟩	1	No	
21	⟨W.,Bush⟩	1	No	
22	⟨Bush,are⟩	1	No	
23	⟨are,43⟩	1	No	
24	⟨43,Presidents⟩	1	No	

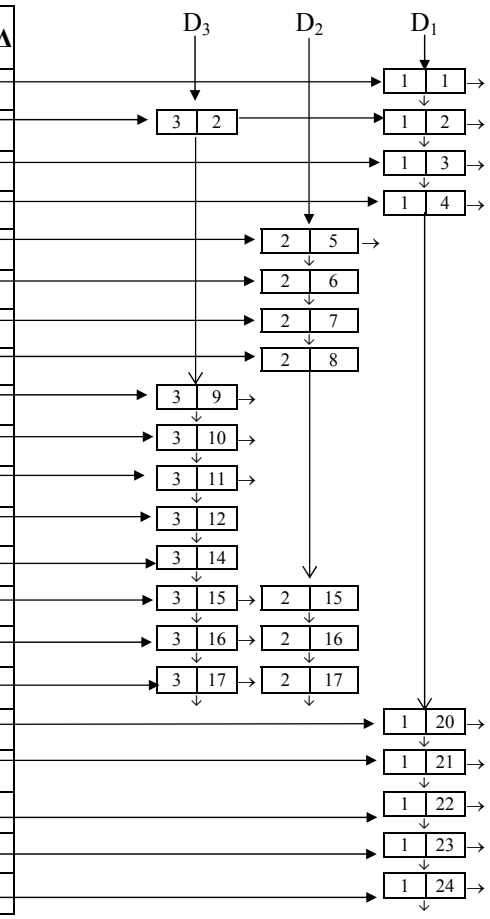


Figura 3.13 Estructura de datos construida para la colección de documentos de la tabla 3.2.

Si n es el número total de palabras en la colección de documentos, se observa en el algoritmo de la etapa 2 que para cada par de palabras consecutivas en la colección de documentos se crea un sólo nodo δ. Si m es el número de pares de palabras diferentes en BDD entonces el arreglo es de tamaño m. Por lo que el algoritmo de la segunda etapa presenta un tiempo $\Theta(n)$ y un espacio $\Theta(n+m)$ donde $m<n$.

3.1.2.3 Tercera etapa, búsqueda de SF's de acuerdo al umbral β

Usando la estructura de datos construida en la etapa anterior y de acuerdo al umbral β, en la tercera etapa Dimasp-C_0 descubre para cada par de palabras la SF más larga que se pueda formar iniciando con ese par, utilizando los documentos de la colección. Esto se hace de la siguiente manera, comenzando con el último documento (D_j) agregado a la estructura de

datos, se verifica cuál de los nodos $\delta(\langle w_i,w_{i+1}\rangle)$ tiene una frecuencia mayor o igual que β ($C_f \geq \beta$), en cuyo caso la *SF* inicial es $\langle w_i,w_{i+1}\rangle$. Cabe recordar que no es necesario buscar todas las ocurrencias del par $\langle w_i,w_{i+1}\rangle$ en la BDD pues se encuentran en *Lista* Δ. Después se debe determinar hasta donde puede crecer la *SF*. Una $SF=\langle w_i,w_{i+1}\rangle$ puede crecer a $\langle w_i,w_{i+1},w_{i+2}\rangle$ si existen al menos β ocurrencias de ésta última en diferentes documentos, lo cual se lleva a cabo verificando cuáles de los nodos $\delta(\langle w_i,w_{i+1}\rangle)$ en *Lista* Δ tienen como siguiente par a $\delta(\langle w_{i+1},w_{i+2}\rangle)$. En cuyo caso, sólo son considerados para el crecimiento de la SF los nodos $\delta(\langle w_{i+1},w_{i+2}\rangle)$, los cuales representan el último par de la secuencia $\langle w_i,w_{i+1},w_{i+2}\rangle$. En general, la palabra w_{i+k} se puede añadir a la *SF* $\langle w_i,w_{i+1},\ldots,w_{i+(k-2)},w_{i+(k-1)}\rangle$, si entre los nodos $\delta(\langle w_{i+(k-2)},w_{i+(k-1)}\rangle)$ utilizados para el crecimiento, existen al menos β que continúen con el par $\delta(\langle w_{i+(k-1)},w_{i+(k)}\rangle)$, en cuyo caso sólo son considerados para el crecimiento de la *SF* los nodos $\delta(\langle w_{i+(k-1)},w_{i+(k)}\rangle)$ que representan el último par de la secuencia $\langle w_i,w_{i+1},\ldots,w_{i+(k)}\rangle$. Si la *SF* no puede crecer más entonces la *SF* es agregada al conjunto de SF's y todas las celdas del arreglo usadas para formar la *SF* son marcadas como "usadas".

De hecho, sólo las SF's con una longitud mayor o igual a tres son agregadas al conjunto de SF's y todas las celdas del arreglo usadas para formar esa *SF* son marcadas como "usadas". Así, una vez que todos los documentos son procesados, las celdas no marcadas como "usadas" y con $C_f \geq \beta$ son agregadas como secuencias frecuentes maximales de longitud 2 (2~*SFM's*). De forma similar, todas las palabras con una frecuencia mayor o igual que β que no fueron usadas en el conjunto de 2~*SFM's* son agregadas como 1~*SFM's*. De esta manera en la etapa 4 se evitan comparaciones para determinar cuáles de las 1~*SF's* y 2~*SF's* son maximales. La figura 3.18 muestra el algoritmo de la tercera etapa, el cual a su vez utiliza el algoritmo de la figura 3.19 para determinar las 1~*SFM's* y 2~*SFM's*.

Por ejemplo, utilizando la estructura de datos de la figura 3.13 y $\beta = 2$, Dimasp-C_0 encuentra para cada par de palabras la SF más larga que se pueda formar utilizando los documentos de la colección. Para esto se encuentran las SF's del documento D_3, luego las del documento D_2 y finalmente las del documento D_1. Para D_3 la primera SF inicia con $\langle George,Washington\rangle$, pues ese par tiene una frecuencia mayor o igual a $\beta = 2$. En este momento sabemos que la *Lista* Δ del par $\langle George,Washington\rangle$ permite recuperar todas sus ocurrencias en la BDD por lo que sólo es necesario determinar hasta donde puede crecer la $SF=\langle George,Washington\rangle$ con los siguientes pares de palabras del documento D_3. En este caso, el siguiente par de palabras es $\langle Washington,was\rangle$, sin embargo solo ocurre en D_3 y es necesario que ocurra en al menos $\beta = 2$ documentos, por lo que la *SF* no puede crecer. La figura 3.14 muestra la parte de la estructura que está involucrada en el crecimiento de la $SF=\langle George,Washington\rangle$. En esta figura se puede apreciar que a partir de la *Lista* Δ sólo el

documento D_3 tiene el par de palabras consecutivo ⟨*Washington,was*⟩ ya que D_1 tiene a ⟨*Washington,to*⟩. Puesto que la *SF* ⟨*George,Washington*⟩ ya no puede crecer más y su longitud es menor a 3, entonces la *SF* no es agregada al conjunto de SF's.

índice	$\langle w_i,w_{i+1}\rangle$	C_f	¿El par fue usado?	Lista Δ
1	⟨From,George⟩	1	No	
2	⟨George,Washington⟩	2	No	
3	⟨Washington,to⟩	1	No	
4	⟨to,George⟩	1	No	
5	⟨Washington,is⟩	1	No	
6	⟨is,the⟩	1	No	
7	⟨the,capital⟩	1	No	
8	⟨capital,of⟩	1	No	
9	⟨Washington,was⟩	1	No	

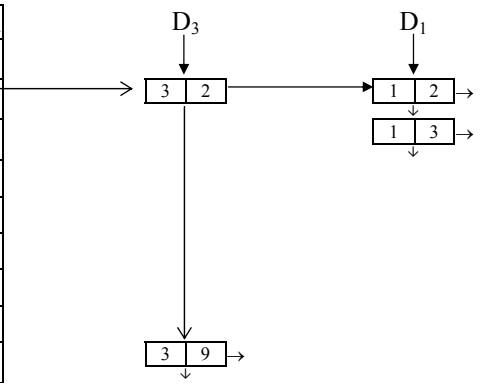


Figura 3.14 Parte de la estructura de datos involucrada en el crecimiento de la secuencia ⟨*George,Washington*⟩ con respecto al documento D_3.

La siguiente SF del documento D_3 inicia con ⟨*of,the*⟩, pues ese par tiene una frecuencia mayor o igual a β. Ahora es necesario determinar hasta donde puede crecer la *SF*=⟨*of,the*⟩ con los siguientes pares de palabras del documento D_3. En este caso, ⟨*the,United*⟩ es el siguiente par de palabras que se presenta tanto en D_2 como en D_3, lo cual permite que la *SF* pueda crecer a ⟨*of,the,United*⟩. Del mismo modo, se puede crecer la *SF* hasta obtener ⟨*of,the,United,States*⟩. Dado que se trata de una 4~*SF*, todas las celdas correspondientes a los pares (⟨*of,the*⟩, ⟨*the,United*⟩, ⟨*United,States*⟩) son marcadas como "usadas". La figura 3.15 muestra parte de la estructura involucrada en el crecimiento de la *SF*=⟨*of,the,United,States*⟩.

índice	$\langle w_i,w_{i+1}\rangle$	C_f	¿El par fue usado?	Lista Δ
15	⟨of,the⟩	2	Si	
16	⟨the,United⟩	2	Si	
17	⟨United,States⟩	2	Si	

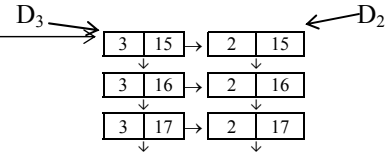


Figura 3.15 Parte de la estructura de datos involucrada en el crecimiento de la secuencia ⟨*of,the,United,States*⟩ con respecto al documento D_3.

De manera similar, este proceso se aplica a los pares de palabras ⟨*the,United*⟩ y ⟨*United,States*⟩, pues tienen una frecuencia mayor o igual a β, produciendo *SF*=⟨*the,United,States*⟩ y *SF*=⟨*United,States*⟩, respectivamente.

Hasta este momento se ha analizado el documento D_3 por lo que se continúa con el documento D_2. En este caso la primera SF inicia con $\langle of,the \rangle$, pues tiene una frecuencia mayor o igual a β. Ahora, a partir de *Lista* Δ se determina hasta donde puede crecer la $SF=\langle of,the \rangle$ con los siguientes pares de palabras del documento D_2. En este caso, $\langle the,United \rangle$ es el siguiente par de palabras que sólo se presenta en D_2, lo cual detiene el crecimiento de la *SF*. La figura 3.16 muestra parte de la estructura involucrada en el crecimiento de la $SF=\langle of,the \rangle$. Nótese que, a diferencia de la figura 3.15, sólo son considerados en *Lista* Δ los nodos δ a partir del documento analizado, es por eso que la SF encontrada $\langle of,the \rangle$ no creció a $\langle of,the,United,States \rangle$ como sucedió con D_3, puesto que D_3 no se considera porque ya fue procesado.

índice	$\langle w_i,w_{i+1} \rangle$	C_f	**¿El par fue usado?**	**Lista Δ**
15	⟨of,the⟩	2	Si	●
16	⟨the,United⟩	2	Si	●
17	⟨United,States⟩	2	Si	●

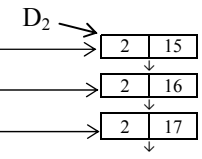


Figura 3.16 Parte de la estructura de datos involucrada en el crecimiento de la secuencia $\langle of,the,United,States \rangle$ con respecto al documento D_2.

Este proceso es aplicado al resto de los documentos (D_2 y D_1) sin producir SF alguna. Finalmente, todas las celdas en el arreglo no marcadas como "usadas" son agregadas como $2\sim SFM$ y todas las palabras frecuentes no contenidas en una celda frecuente son agregadas como $1\sim SFM$. De esta manera, la secuencia $\langle George,Wasihngton \rangle$ es descubierta como $2\sim SFM$. Cabe señalar que las ligas de las figuras 3.14, 3.15 y 3.16 fueron modificadas para efectos de visualización, sin embargo las ligas no son modificadas durante esta etapa.

El peor caso de la tercera etapa sucede cuando se produce el mayor número de SF's con la mayor longitud posible. Este caso en particular ocurre cuando la colección está formada por p documentos idénticos, de m palabras diferentes y con el umbral más bajo $\beta = 2$. En tal caso, todas las m celdas del arreglo son frecuentes y para cada celda el crecimiento de la *SF* utilizará los p documentos de la colección llegando a crecer hasta m, es decir, se producirán mp secuencias frecuentes hasta de longitud m. De esta manera, el peor tiempo sería de orden $O(pm^2)$. Cabe señalar que esta cota indica el máximo número de SF's que se pueden generar, sin embargo en este caso solamente se generaría una sola SFM de longitud m. En la figura 3.17 se muestra un ejemplo del tipo de estructura de datos construida para el peor caso de la tercera etapa.

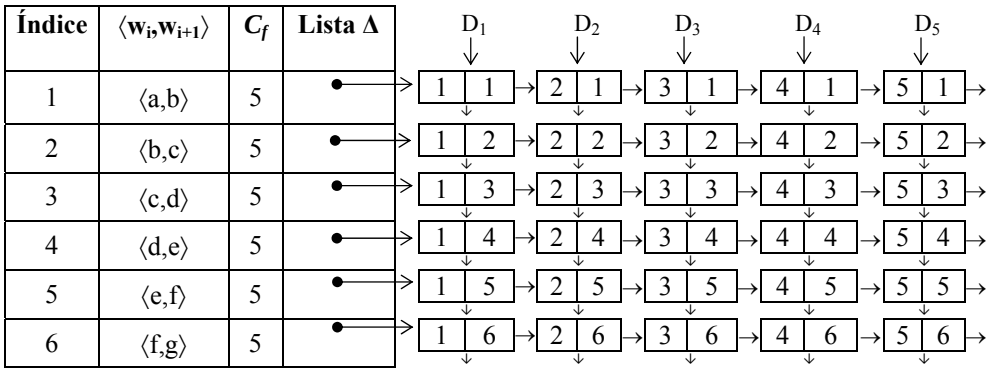


Figura 3.17 Estructura de datos con cinco documentos iguales: ⟨*a,b,c,d,e,f,g*⟩.

Etapa 3: Algoritmo para encontrar todas las SF's
Entrada: Estructura de la etapa 2 y un umbral β **Salida:** Conjunto de SF's
Para *todos los documentos* $D_j \in BDD$ **hacer**
Conjunto de SF's ← Encuentra todas las SF's con respecto al documento (D_j)
fin-para
Etapa 3.1: Algoritmo para encontrar todas las SF's con respecto al documento $\mathbf{D_j}$
Entrada: Un documento D_j de la estructura de datos y un umbral β
Salida: El conjunto de SF's con respecto a D_j
Para *todos los nodos* $\delta_i(\langle w_i,w_{i+1}\rangle) \in D_j$ **hacer**
Si Arreglo $[\delta_i.index].frecuencia \geq \beta$ **entonces**
$SF \leftarrow$ Arreglo $[\delta_i.index].\langle w_i,w_{i+1}\rangle$ *//la SF inicial es el par* $\langle w_i,w_{i+1}\rangle$
Lista $\Delta' \leftarrow \{ \delta_i(\langle w_i,w_{i+1}\rangle) \in Arreglo[\delta_i.index].Lista\ \Delta \}$
Lista $\Delta'.frecuencia \leftarrow$ Número de documentos diferentes en Δ' *calculado con* $\delta.Id$
$k \leftarrow 2$
Mientras *Lista* $\Delta'.frecuencia \geq \beta$ **hacer** *el crecimiento de la SF*
Lista $\Delta' \leftarrow \{ \delta_{i+(k-1)} \in Lista\ \Delta' \mid \delta_{i+(k-2)}(\langle w_{i+(k-2)},w_{i+(k-1)}\rangle) \rightarrow \delta_{i+(k-1)}(\langle w_{i+(k-1)},w_{i+(k)}\rangle) \}$
Lista $\Delta'.frecuencia \leftarrow$ Número de documentos diferentes en Δ' *calculado con* $\delta.Id$
Si *Lista* $\Delta'.frecuencia \geq \beta$ **entonces** *puede crecer la SF*
$(i+k)\sim SF \leftarrow \langle w_i,w_{i+1},\ldots,w_{i+(k-2)},w_{i+(k-1)},w_{i+(k)}\rangle$
Arreglo $[\langle w_{i+(k-1)},w_{i+(k)}\rangle].marca \leftarrow$ "usada"
$k \leftarrow k+1$
fin-si
fin-mientras
Si *para* $(i+k)\sim SF$, $(i+k) \geq 3$ **entonces** *agregar la SF al conjunto de SF's*
SF's ← *agregar la* $(i+k)\sim SF$
fin-si
fin-para

Figura 3.18 Algoritmo para determinar de cada par de palabras la SF más larga que se pueda formar usando la estructura de datos construida en la segunda etapa.

Etapa 3.2: Algoritmo para encontrar todas las* 1~*SFM's y* 2~*SFM's

Entrada: Umbral β; Arreglo de *Palabras* con su frecuencia (etapa 1); *Arreglo* de pares de palabras (etapa 3.1)

Salida: Conjunto de 1~SFM's y 2~SFM's

Para *cada celda i del arreglo Pares de palabras* **hacer**

Si *Arreglo*[*i*].*frecuencia* $\geq \beta$ *y Arreglo*[*i*].*marca= "No usada"* **entonces** *se trata de una* 2~*SFM*

{2~*SFM's*} ←*Arreglo*[*i*]. $\langle w_i, w_{i+1} \rangle$

Palabras [*w*].*marca* ← *Usada*

Palabras [w_{i+1}].*marca* ←*Usada*

fin-para

Para *cada celda i del arreglo de Palabras* **hacer**

Si *Palabras*[*i*].*frecuencia* $\geq \beta$ *y Palabras*[*i*].*marca= "No usada"* **entonces** *se trata de un* 1~*SFM*

1~*SFM* ←*Palabras*[*i*]**.** $\langle w_i \rangle$

fin-para

Figura 3.19 Algoritmo para encontrar todas las SFM's de longitud uno y dos.

3.1.2.4 Cuarta etapa, selección de las SF's maximales a partir del conjunto de SF's

A partir de todo el conjunto de secuencias frecuentes encontradas en la etapa 3 se pueden seleccionar las maximales de manera rápida. Para este propósito se utiliza parte de la idea de un árbol de sufijos (para ver qué es un árbol de sufijos, ver la sección 2.2.6).

Utilizando un árbol de sufijos es posible determinar, del conjunto de SF's encontradas en la etapa 3, aquellas que son maximales. Sin embargo, para este propósito se modificó la construcción del árbol de manera que al terminar de analizar cada SF sea posible determinar cuales son las maximales. En este caso, para cada SF analizada se agrega al árbol únicamente el primer sufijo (es decir la SF completa) y los demás sufijos sólo se utilizan para eliminar secuencias previamente agregadas al árbol que no sean maximales. Es por eso que las SF's son analizadas de menor a mayor longitud. De esta manera, al terminar de construir el árbol solamente quedarán las SFM's, una en cada rama. A continuación se enlistan todas las SF's de longitud mayor a 2 encontradas por la tercera etapa con la estructura de datos de la figura 3.21:

- ⟨*the, United, States*⟩
- ⟨*George, W. , Bush*⟩
- ⟨*of ,the, United, States*⟩
- ⟨*President, of, the, United, States*⟩

Por ejemplo, a partir de las SF's anteriores se muestra en la figura 3.20 la forma en cómo se construye el árbol de SFM's en la cuarta etapa. En la figura 3.20(a) se muestra el árbol de SFM's después de agregar el primer sufijo de la SF ⟨*the, United, States*⟩ (es decir la SF completa). La figura 3.20(b) muestra el árbol de SFM's después de agregar el primer sufijo de la SF ⟨*George, W. , Bush*⟩. La figura 3.20(c) muestra el árbol de SFM's después de agregar el primer sufijo de la SF ⟨*of ,the, United, States*⟩. Sin embargo, en este caso el segundo sufijo ⟨*the, United, States*⟩ elimina la rama de la primera secuencia ⟨*the, United, States*⟩, puesto que son iguales y por lo tanto no es maximal, lo cual se muestra en la figura 3.20(d). En la figura 3.20(e) se muestra el árbol de SFM's después de agregar la SF ⟨*President, of, the, United, States*⟩, también en este caso el segundo sufijo elimina a una rama del árbol, de esta manera el árbol de SFM's resultante de la cuarta etapa es el de la figura 3.20(f).

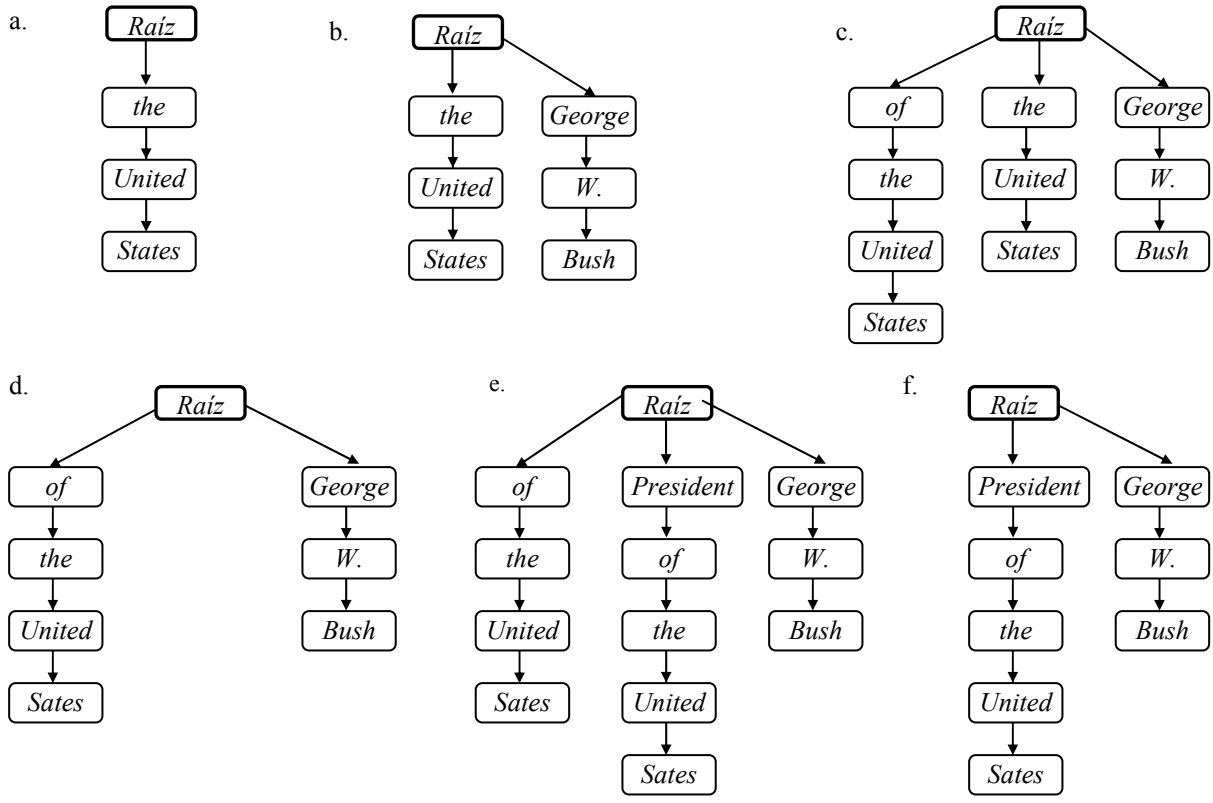


Figura 3.20 Ejemplo de cómo agregar y procesar las SF's en el árbol de SFM's.

Para esta etapa el peor caso en espacio se presenta cuando cada una de las SF's producidas en la tercera etapa tiene su primera palabra diferente y constituye una SFM, de

manera que por cada SFM se tiene una rama en el árbol. Si n es el número total de palabras en la BDD entonces en Dimasp-C_0 se pueden encontrar a lo más $n-1$ pares de palabras diferentes, entonces se pueden formar a lo más n SFM's por lo que el árbol presenta un espacio $O(n)$. Cabe señalar que la cota dada en la tercera etapa $O(pm^2)$ indica el máximo número de SF's que se pueden generar, sin embargo en ese caso solamente se generaría una sola SFM de longitud m por lo que sería uno de los mejores casos para la cuarta etapa.

Para revisar si una SF es maximal puede ser necesario revisar en el árbol a todos los sufijos de cada SF. En la tercera etapa de Dimasp-C_0 se indicaba que en el peor caso se pueden producir a lo más pm secuencias frecuentes hasta de longitud m, sin embargo en la cuarta etapa puede ser necesario recorrer los m sufijos de cada SF encontrada, por lo que la cuarta etapa es de $O(pm^3)$.

Para probar que nuestro algoritmo encuentra todas las SFM's se introduce la siguiente proposición.

Proposición 1: *Dimasp-C_0 descubre todas las secuenciales frecuentes maximales de una BDD.*
Prueba. Para probar que Dimasp-C_0 encuentra todas las SFM's, mostraremos que si S es una k~SFM entonces S puede ser construida a partir de la estructura de datos. Dado que S es una k~SFM = $\langle w_1, w_2, w_3, \ldots, w_k \rangle$ entonces sabemos que está contenida en al menos β documentos en la base de documentos. Además se sabe que todos los pares de palabras $\langle w_1, w_2 \rangle$, $\langle w_2, w_3 \rangle, \cdots$, $\langle w_{k-1}, w_k \rangle$ de la k~SFM también son frecuentes. Por la manera en que trabaja Dimasp-C_0 sabemos que en la estructura de datos se pueden localizar las ocurrencias del par $\langle w_1, w_2 \rangle$ y en al menos β ocurrencias sigue el par $\langle w_2, w_3 \rangle$ y esto puede continuar así hasta el par $\langle w_{k-1}, w_k \rangle$, y no puede continuar a $\langle w_k, w_{k+1} \rangle$ pues dejaría de ser una k~SFM.

Si k=1 entonces Dimasp-C_0 incluye esta 1~SFM porque en la etapa 1, Dimasp-C_0 incluyó todas las palabras frecuentes las cuales no son incluidas en alguna otra SFM.

Como esto se pudo hacer para S y S puede ser cualquier k~SFM entonces es posible encontrar todas las SFM's de la BDD.■

3.1.2.5 Procesamiento incremental de documentos con Dimasp-C_0

Con el objetivo de no repetir todo el trabajo realizado para descubrir todas las SFM's cuando un conjunto de documentos es agregado a la base de documentos, se usa la etapa 2.1 de

Dimasp-C_0 para agregar los nuevos documentos a la estructura de datos. Después, la etapa 3.1 es aplicada únicamente sobre los nuevos documentos con lo que se descubre un conjunto nuevo de SF's. Utilizando el conjunto previo de SFM's y el conjunto nuevo de SF's se determina cuáles son las maximales, para esto se utiliza el algoritmo de la etapa 4.

Por ejemplo, la figura 3.21 muestra la estructura de datos después de agregar el documento D_4="*the President of the United States is George W. Bush*". Usando $\beta = 2$ en el algoritmo de la etapa 3 se descubren las SFM's ⟨*President,of,the,United,States*⟩ y ⟨*George,W.,Bush*⟩. Por lo que la SFM ⟨*President,of,the,United,States*⟩ elimina a la SF previamente descubierta ⟨*of,the,United,States*⟩ porque dejó de ser maximal.

3.1.2.6 Búsqueda de secuencias frecuentes con diferente umbral β

En el capítulo dos se puede ver que los algoritmos que permiten encontrar SFM's con restricción GAP desechan todo el trabajo realizado una vez encontrado el conjunto de SFM's para un umbral β dado. En Dimasp-C_0 no sucede así, pues si se quiere calcular las SFM's para otro valor de β, sólo las etapas tres y cuatro deben ser aplicadas, sin reconstruir la estructura de datos.

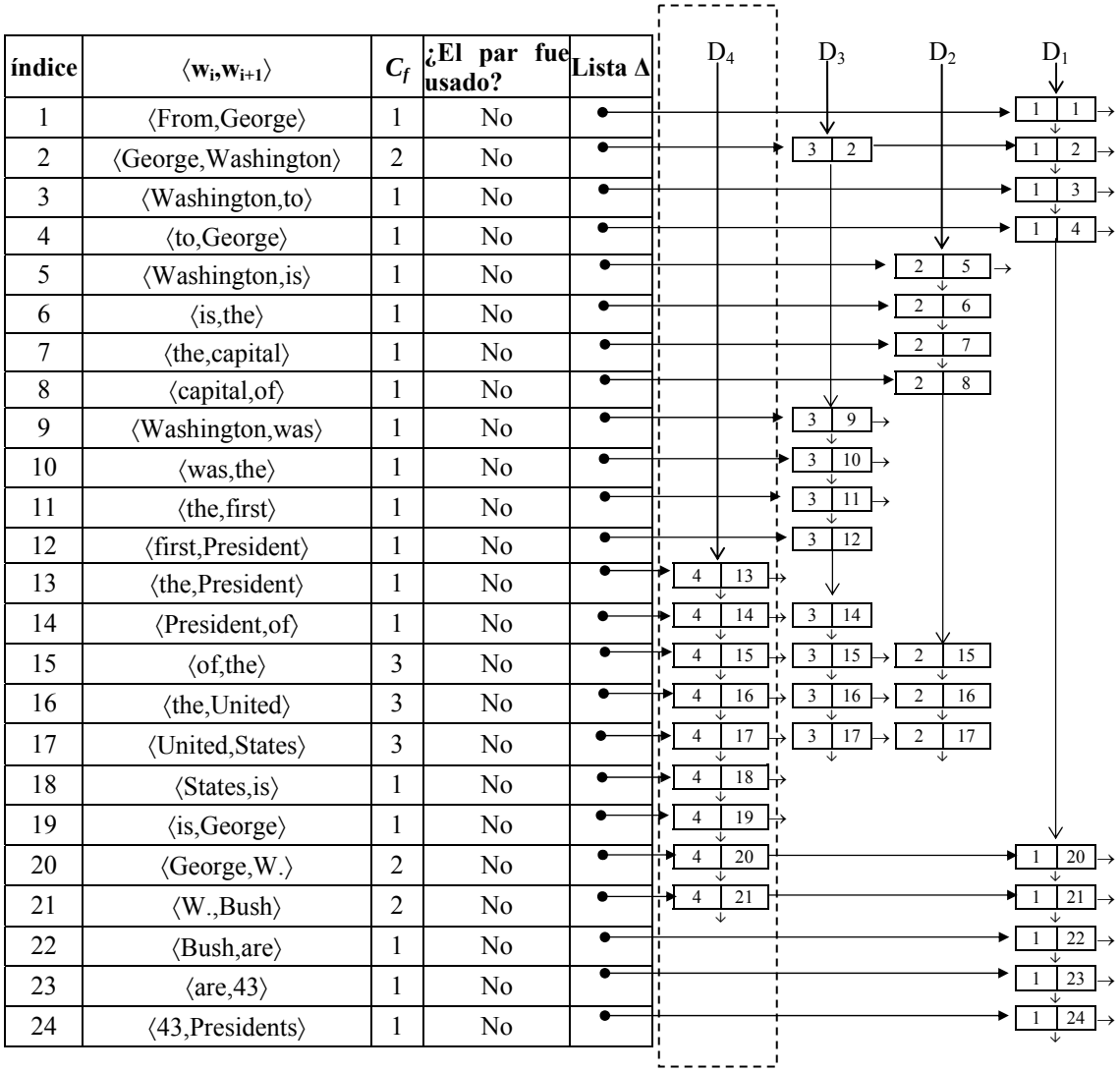


Figura 3.21 Estructura de datos construida para la colección de documentos de la tabla 3.1, después de agregar el documento D_4= "*the President of the United States is George W. Bush"*.

3.1.3. Experimentación

A continuación se presenta una serie de experimentos con el objetivo de mostrar el desempeño de Dimasp-C_0 frente a otros algoritmos cuando son ejecutados bajo las mismas condiciones, esto es, con la misma computadora y con la misma colección de documentos.

También se muestran una serie de experimentos con el objetivo de determinar el desempeño de Dimasp-C_0 cuando un conjunto de documentos es agregado después de haber procesado una colección de documentos. Además se incluyen algunas gráficas donde es posible observar la distribución del conjunto de SFM's encontradas.

Los experimentos se llevaron a cabo usando la colección de documentos Reuters-21578 [LEWIS] la cual contiene 21578 noticias. Después de eliminar 226 *stop-words*[1] (ver anexo 1), la colección tiene 39,688 palabras diferentes de 1.18 millones de palabras usadas en toda la colección. Después de eliminar las *stop-words* la longitud promedio de los documentos es de 63 palabras. En todos los experimentos fueron usados los primeros 5000, 10000, 15000 y 20000 documentos. En los experimentos se utilizaron los programas originales proporcionados por los autores, excepto para el algoritmo GSP el cual fue programado por nosotros. Dimasp-C_0 fue programado en Builder C++ versión 6. Los experimentos aquí mostrados fueron ejecutados en una computadora Pentium 4 a 3Ghz con 1GB de RAM.

En la figura 3.22 se muestra la comparación de todos los algoritmos que permiten obtener SFM's con restricción GAP=0 usando un β =15: Dimasp-C_0, cSPADE [Mohammed 2000a], GenPrefixSpan [Antunes 2003], DELISP [Ming 2002] y GSP [Agrawal 1996]. En este caso, GenPrefixSpan tuvo problemas con la memoria principal por lo que sólo fue posible probar con los primeros 5000 y 10000 documentos.

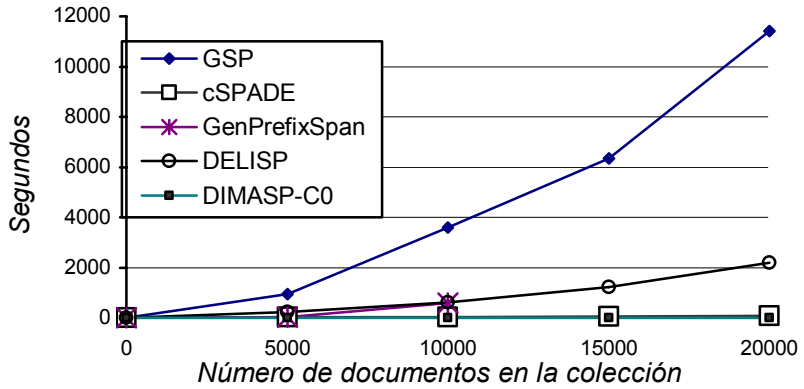


Figura 3.22 Comparación de ejecución con $\beta = 15$.

[1] Palabras consideradas como vacías, tales como: conjunciones, preposiciones etc.; la cuales generalmente no son útiles para tareas de procesamiento automático de texto.

La figura 3.22 permite observar como GSP, GenPrefixSpan y DELISP muestran un desempeño inferior que cSPADE y Dimasp-C_0. La figura 3.23 muestra la misma comparación de la figura 3.22 pero eliminando el peor algoritmo (GSP), aquí es posible ver que Dimasp-C_0 fue mejor que cSPADE, aunque se mantienen cercanos.

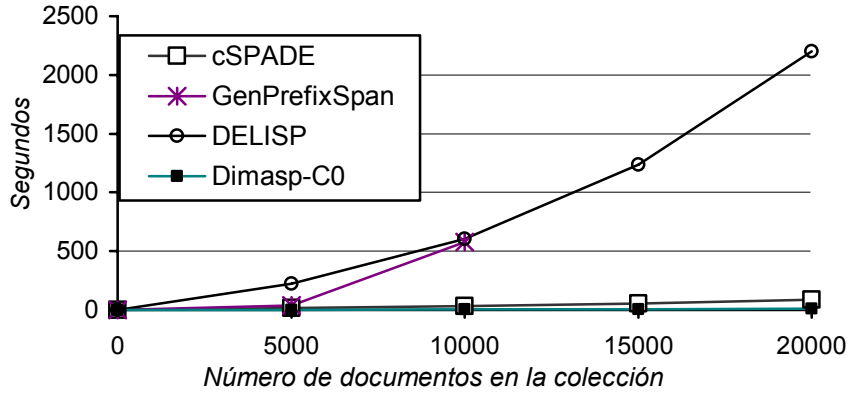


Figura 3.23 Comparación del desempeño de Dimasp-C_0 con $\beta = 15$.

En la figura 3.24 se compara el desempeño de Dimasp-C_0 contra el algoritmo cSPADE, que es el más rápido de los algoritmos previos (según nuestros experimentos). Aquí puede notarse que Dimasp-C_0 supera notablemente a cSPADE.

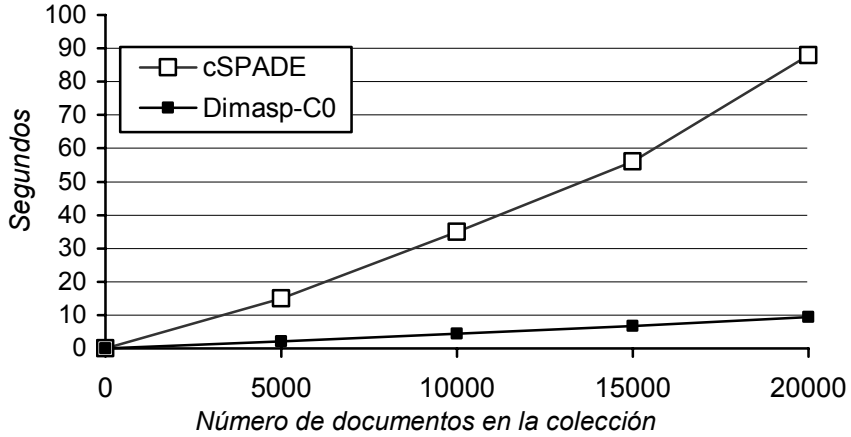


Figura 3.24 Comparación de Dimasp-C_0 contra cSPADE con $\beta = 15$.

La figura 3.25 muestra el comportamiento de Dimasp-C_0 para diferentes β's, en esta figura se aprecia que el umbral de frecuencia no afectó en gran medida el tiempo de ejecución.

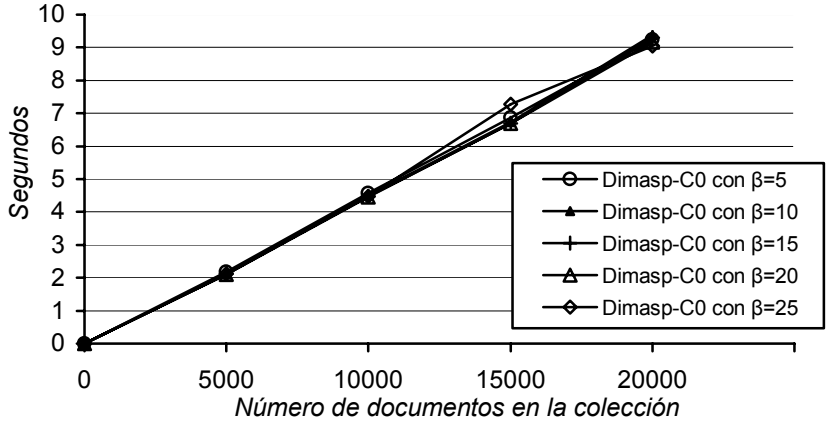


Figura 3.25 Comparación de Dimasp-C_0 con β = 5, 10, 15, 20 y 25.

La figura 3.26 muestra el tiempo acumulado por las etapas de Dimasp-C_0 con β = 15 y permite observar que de todas las etapas, la primera etapa (donde se transforman los documentos) consume el mayor tiempo de procesamiento cuando β = 15, y esto se cumple para diferentes tamaños de la colección de documentos. Sin embargo, la primera etapa es independiente del umbral de frecuencia por lo que sólo se tiene que procesar una sola vez cuando el usuario requiere procesar diferentes valores de β.

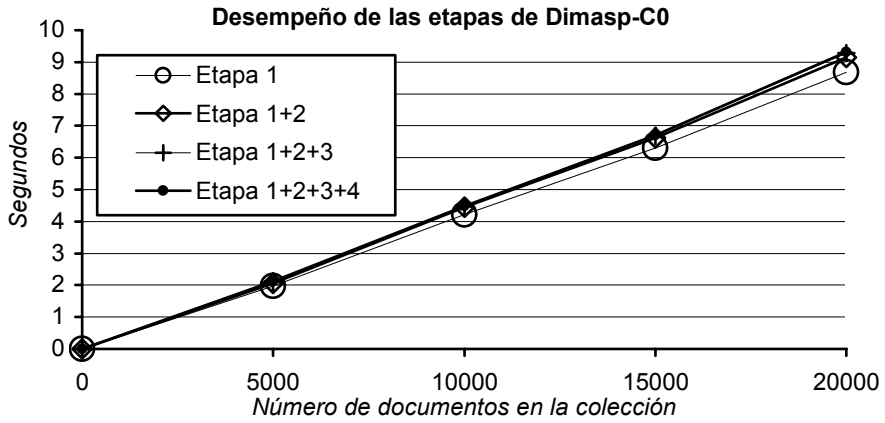


Figura 3.26 Desempeño de las etapas de Dimasp-C_0 con β = 15.

Se realizó un experimento adicional con el β más bajo (β=2), en este experimento Dimasp-C_0 encontró una SFM de longitud 398 pues se descubrió que había dos documentos iguales, la gráfica de figura 3.27 muestra el tiempo de procesamiento.

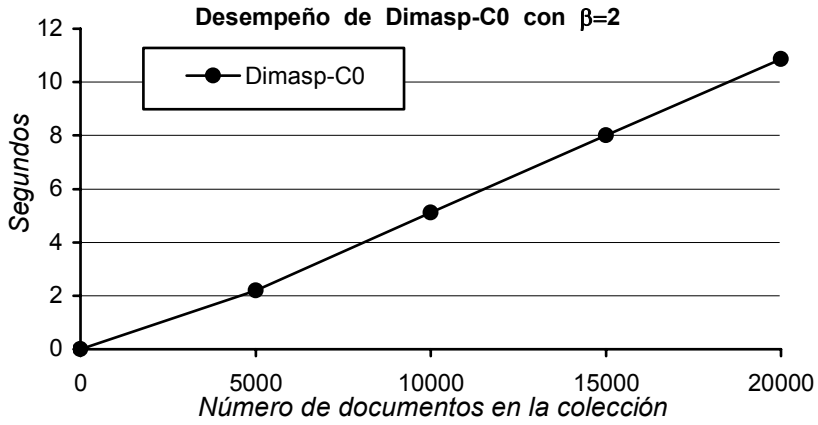


Figura 3.27 Dimasp-C_0 con el umbral más bajo ($\beta = 2$).

Para evaluar el desempeño incremental de Dimasp-C_0 se procesaron 4000, 9000, 14000 y 19000 documentos. Posteriormente se agregaron 1000 documentos en cada experimento. La figura 3.28 muestra la comparación de Dimasp-C_0 contra cSPADE el cual necesita recalcular todas las SFM's cuando se agregan 1000 documentos.

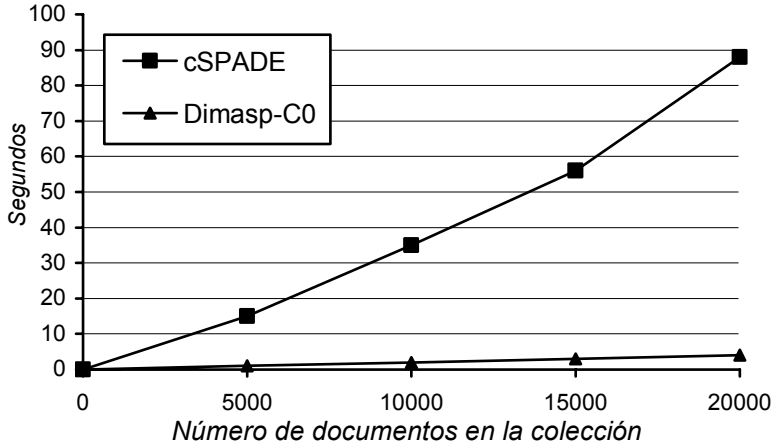


Figura 3.28 Escabilidad incremental de Dimasp-C_0 y cSPADE usando $\beta = 15$.

La figura 3.29 muestra la distribución de las SFM's de acuerdo a su longitud, cuando se calcula con $\beta = 15$ sobre los primeros 20000 documentos. Esta gráfica es importante porque muestra donde se encuentran concentradas, respecto a su longitud, el mayor número de SFM's. En este experimento la SFM más grande tiene una longitud de 25, de la cual puede

verse, en el anexo 2, una de las noticias donde aparece dicha SFM. En el anexo 3 se muestran algunas de las SFM's obtenidas en el experimento de la figura 3.29

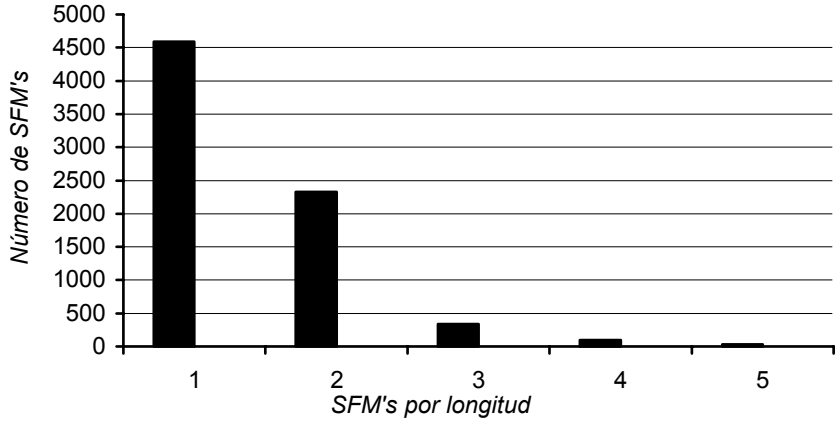


Figura 3.29 Distribución de los k~SFM's con $k \leq 5$ para 20000 documentos con $\beta = 15$.

Finalmente, la figura 3.30 muestra el número de SFM's cuando β es igual al 0.1% de la cantidad de documentos en la colección. En los experimentos anteriores se utilizó un umbral fijo, sin embargo si tomamos el umbral como el 0.1% de la cantidad de documentos en la colección se observa que el número de SFM's varía relativamente poco.

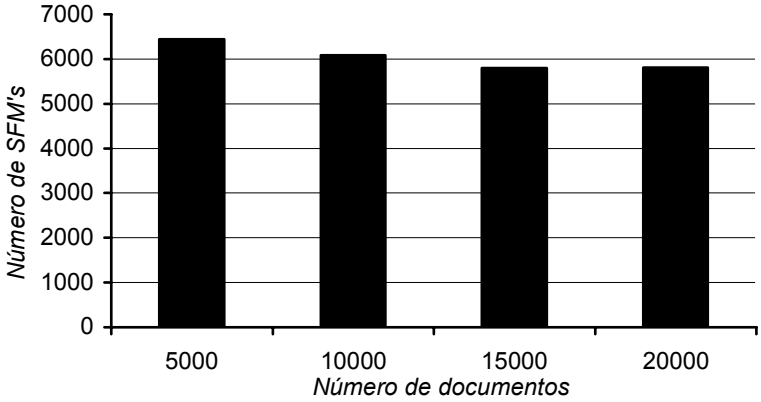


Figura 3.30 Número de SFM's con $\beta = 0.1\%$ con respecto a la cantidad de documentos en BDD.

3.1.4. Sumario

En esta sección se presentó el algoritmo Dimasp-C_0, el cual es un algoritmo diseñado para descubrir todas las secuencias frecuentes maximales con GAP=0 mediante el crecimiento de

patrones. Para esto, Dimasp-C_0 construye una estructura de datos para la colección de documentos a analizar, la cual permite hacer el descubrimiento de las SFM's de manera rápida. Una característica importante es que Dimasp-C_0 permite trabajar con diferentes umbrales de soporte sin tener que reconstruir la estructura de datos. Además, Dimasp-C_0 puede procesar los documentos de manera incremental.

De acuerdo a evaluaciones experimentales, Dimasp-C_0 es más rápido que los algoritmos GSP, DELISP, GenPrefixSpan y cSPADE en el descubrimiento de todas las SFM's en una colección de documentos, permitiendo además calcular las SFM's para diferentes valores de β sin tener que reconstruir la estructura. Una de las razones de por qué Dimasp-C_0 es más eficiente, es porque primero busca las SFM's de longitud mayor a igual a 3, y las SFM's de longitud menor a 3, que son la mayoría, son descubiertas en una sola pasada. Por ejemplo, la figura 3.18 muestra que el número de SFM's de longitud 1 y 2 es mayor que la suma del resto de las SFM's. En los experimentos se usó la colección Reuters-21578 para la cual Dimasp-C_0 necesitó 30 Mbytes de memoria RAM para la estructura de datos construida en la segunda etapa, lo cual es manejable en las computadoras actuales. Esto también nos indica que, aunque Dimasp-C_0 necesita mantener la estructura de datos en memoria RAM, es factible el procesamiento de colecciones más grandes.

La búsqueda de SFM's con GAP=0, ó lo que es lo mismo, SFM's contiguas o consecutivas, es una herramienta importante para el análisis de diferentes tipos de información como lo son registros de visitas de páginas WEB (mining web logs) [Bamshad 2002][Youssefi 2004], la compresión de datos [Osslan 2005] y en el análisis de texto [Denicia 2006][Villatoro 2006] [Coyotl 2006, 2007][Denicia 2007].

En la siguiente sección se extenderá la idea básica de Dimasp-C_0 para permitir mediante el parámetro GAP controlar la separación máxima que puede haber entre los elementos que forman una SFM.

3.2. Descubrimiento de SFM's en una colección de documentos para GAP=n

En la sección 3.1 se presentó el algoritmo Dimasp-C_0, el cual permite el descubrimiento de todas las SFM's con GAP=0. En esta sección se presenta al algoritmo Dimasp-C_n el cual permite hacer el descubrimiento de todas las SFM's para GAP=n. De esta manera, Dimasp-C_n puede calcular las mismas secuencias que Dimasp-C_0 cuando n=0, sin embargo Dimasp-C_0

lo hace de manera más eficiente. Primero se plantea el problema de búsqueda de SFM's para GAP=n a partir de una colección de documentos. Posteriormente se presenta el algoritmo Dimasp-C_n. Para finalizar, se presenta la experimentación y el sumario.

3.2.1. Definición del problema para GAP=n

Una ***secuencia*** S, denotada por $\langle s_1,s_2,...,s_k\rangle$, es una lista ordenada de k elementos. Una secuencia de ***longitud*** k es denominada k~secuencia.

Sean $P=\langle p_1,p_2,...,p_t\rangle$ y $S=\langle s_1,s_2,...,s_m\rangle$ secuencias, P es una subsecuencia con GAP=n de S, denotado como $P\subseteq_n S$, si existen enteros $1\leq i_1<i_2<...<i_t\leq m$ tal que $p_1 = s_{i_1}, p_2 = s_{i_2}, p_3 = s_{i_3},...,p_t = s_{i_t}$, donde $(i_k - i_{k-1}) - 1 \leq n$. Es decir, n es la máxima separación permitida entre los elementos que pueden formar una subsecuencia.

Un documento W se puede considerar como una secuencia de palabras, denotado también como $\langle w_1,w_2,\ldots,w_t\rangle$.

Dada una colección de documentos $\{W_1, W_2, ..., W_t\}$ y una restricción GAP=n, la ***frecuencia*** de una secuencia S, denotada por S_f, es el número de documentos diferentes donde S es una subsecuencia con GAP=n, esto es , $S_f = |\{ W_i \mid S \subseteq_n W_i \}|$. En esta sección se usará el término ***secuencia*** para referirnos a las ***secuencias con GAP=n.***

Dado un umbral de frecuencia definido por el usuario (β), una secuencia S es ***frecuente*** si $S_f \geq \beta$. Una secuencia frecuente S es ***maximal*** si S no es subsecuencia con GAP=0 (n=0) de alguna otra secuencia frecuente. A una secuencia frecuente maximal (SFM) también se le conoce como ***patrón secuencial maximal***.

Dada una colección de documentos, el problema abordado en esta sección es el descubrimiento de todas las secuencias frecuentes maximales con GAP=n para un umbral β dado.

3.2.2. Algoritmo propuesto para GAP=n (Dimasp-C_n)

Al igual que Dimasp-C_0, Dimasp-C_n utiliza una estructura de datos para el descubrimiento de todas las SFM's con GAP=n. Para la construcción de la estructura de datos Dimasp-C_n representa cada uno de los documentos mediante los pares de palabras que el GAP permite. Por ejemplo, si tuviéramos GAP=1 entonces el documento D_3=⟨*a,b,c,d*⟩ se puede representar por los pares de palabras ⟨*a,b*⟩, ⟨*b,c*⟩, ⟨*c,d*⟩, ⟨*a,c*⟩ y ⟨*b,d*⟩. Note que ⟨*a,c*⟩ es un par que permite el GAP, puesto que la separación en el documento entre "a" y "c" es mayor o igual a GAP. En la tabla 3.3 se muestra parte de la representación utilizada por Dimasp-C_n para la colección de documentos de la tabla 3.1, pero ahora GAP=1. Si se comparan las tablas 3.1 y 3.3 se puede apreciar que en la tabla 3.3 sólo fueron agregados los pares de palabras que permite el GAP=1.

Tabla 3.3 Ejemplo de una colección de documentos y de su representación mediante pares de palabras. El subíndice en los pares de palabras representa la frecuencia de ese par de palabras en la colección de documentos.

Documento	Documento original	Representación del documento mediante sus pares de palabras que GAP=1 permite
D_1	⟨*a,b,f*⟩	$\langle a,b\rangle_4\ \langle a,f\rangle_2\ \langle b,f\rangle_2$
D_2	⟨*b,c,d*⟩	$\langle b,c\rangle_5\ \langle b,d\rangle_3\ \langle c,d\rangle_3$
D_3	⟨*a,b,c,d*⟩	$\langle a,b\rangle_4\ \langle a,c\rangle_3\ \langle b,c\rangle_5\ \langle b,d\rangle_3\ \langle c,d\rangle_3$
D_4	⟨*a,b,c,d,e,a,b,f*⟩	$\langle a,b\rangle_4\ \langle a,c\rangle_3\ \langle b,c\rangle_5\ \langle b,d\rangle_3\ \langle c,d\rangle_3\ \langle c,e\rangle_3\ \langle d,e\rangle_1$ $\langle d,a\rangle_1\ \langle e,a\rangle_1\ \langle e,b\rangle_1\ \langle a,b\rangle_4\ \langle a,f\rangle_2\ \langle b,f\rangle_2$
D_5	⟨*b,c,e*⟩	$\langle b,c\rangle_5\ \langle b,e\rangle_2\ \langle c,e\rangle_3$
D_6	⟨*a,b,c,e*⟩	$\langle a,b\rangle_4\ \langle a,c\rangle_3\ \langle b,c\rangle_5\ \langle b,e\rangle_2\ \langle c,e\rangle_3$

Mediante la representación de documentos presentada en la tabla 3.3 es posible encontrar las SFM's con GAP=n. Para mostrar cómo se encuentra una SFM, supongamos que el umbral de frecuencia β es 2 y GAP es igual a 1, por lo que en la colección de documentos de la tabla 3.3, ⟨*a,b,c,e*⟩ es una *SFM*. En este caso, la secuencia ⟨*a,b,c,e*⟩ también puede ser representada mediante los pares de palabras $\langle a,b\rangle_4$, $\langle b,c\rangle_5$ y $\langle c,e\rangle_3$, los cuales tienen cada uno una frecuencia mayor o igual a β. Esto nos permite saber que es posible localizar la *SFM* si encontramos primero todas las ocurrencias de la secuencia ⟨*a,b*⟩ en toda la colección de documentos. En la tabla 3.4 están señaladas las ocurrencias del par de palabras $\langle a,b\rangle_4$ de la colección de documentos de la tabla 3.3.

Tabla 3.4 Ocurrencias del par de palabras $\langle a,b \rangle$ en una colección de documentos.

Documento	Documento original	Representación del documento mediante sus pares de palabras que GAP=1
D_1	$\langle \underline{a,b},f \rangle$	$\mathbf{\langle a,b \rangle_4}\ \langle a,f \rangle_2\ \langle b,f \rangle_2$
D_2	$\langle b,c,d \rangle$	$\langle b,c \rangle_5\ \langle b,d \rangle_3\ \langle c,d \rangle_3$
D_3	$\langle \underline{a,b},c,d \rangle$	$\mathbf{\langle a,b \rangle_4}\ \langle a,c \rangle_3\ \langle b,c \rangle_5\ \langle b,d \rangle_3\ \langle c,d \rangle_3$
D_4	$\langle \underline{a,b},c,d,e,\underline{a,b},f \rangle$	$\mathbf{\langle a,b \rangle_4}\ \langle a,c \rangle_3\ \langle b,c \rangle_5\ \langle b,d \rangle_3\ \langle c,d \rangle_3\ \langle c,e \rangle_3\ \langle d,e \rangle_1$ $\langle d,a \rangle_1\ \langle e,a \rangle_1\ \langle e,b \rangle_1\ \mathbf{\langle a,b \rangle_4}\ \langle a,f \rangle_2\ \langle b,f \rangle_2$
D_5	$\langle b,c,e \rangle$	$\langle b,c \rangle_5\ \langle b,e \rangle_2\ \langle c,e \rangle_3$
D_6	$\langle \underline{a,b},c,e \rangle$	$\mathbf{\langle a,b \rangle_4}\ \langle a,c \rangle_3\ \langle b,c \rangle_5\ \langle b,e \rangle_2\ \langle c,e \rangle_3$

Una vez encontrado el conjunto de todas las ocurrencias de la secuencia $\langle a,b \rangle$, habrá que descartar aquellas ocurrencias que no continúen con el par de palabras $\langle b,c \rangle$ de acuerdo a la separación permitida por el GAP. De esta manera el nuevo conjunto de ocurrencias corresponde a la secuencia $\langle a,b,c \rangle$, lo cual se muestra en la tabla 3.5.

Tabla 3.5 Ocurrencias de la secuencia de palabras $\langle a,b,c \rangle$ en una colección de documentos.

Documento	Documento original	Representación del documento mediante sus pares de palabras que GAP=1
D_1	$\langle a,b,f \rangle$	$\langle a,b \rangle_4\ \langle a,f \rangle_2\ \langle b,f \rangle_2$
D_2	$\langle b,c,d \rangle$	$\langle b,c \rangle_5\ \langle b,d \rangle_3\ \langle c,d \rangle_3$
D_3	$\langle \underline{a,b,c},d \rangle$	$\mathbf{\langle a,b \rangle_4}\ \langle a,c \rangle_3\ \mathbf{\langle b,c \rangle_5}\ \langle b,d \rangle_3\ \langle c,d \rangle_3$
D_4	$\langle \underline{a,b,c},d,e,a,b,f \rangle$	$\mathbf{\langle a,b \rangle_4}\ \langle a,c \rangle_3\ \mathbf{\langle b,c \rangle_5}\ \langle b,d \rangle_3\ \langle c,d \rangle_3\ \langle c,e \rangle_3\ \langle d,e \rangle_1$ $\langle d,a \rangle_1\ \langle e,a \rangle_1\ \langle e,b \rangle_1\ \langle a,b \rangle_4\ \langle a,f \rangle_2\ \langle b,f \rangle_2$
D_5	$\langle b,c,e \rangle$	$\langle b,c \rangle_5\ \langle b,e \rangle_2\ \langle c,e \rangle_3$
D_6	$\langle \underline{a,b,c},e \rangle$	$\mathbf{\langle a,b \rangle_4}\ \langle a,c \rangle_3\ \mathbf{\langle b,c \rangle_5}\ \langle b,e \rangle_2\ \langle c,e \rangle_3$

De manera similar, una vez encontrado el conjunto de ocurrencias de la secuencia $\langle a,b,c \rangle$, habrá que descartar aquellas ocurrencias que no continúen con el par de palabras $\langle c,e \rangle$ de acuerdo al GAP. De esta manera el nuevo conjunto de ocurrencias permite encontrar la SFM $\langle a,b,c,e \rangle$ en la colección de documentos, lo cual se muestra en la tabla 3.6.

Tabla 3.6 Ocurrencias de la secuencia de palabras $\langle a,b,c,e\rangle$ en una colección de documentos.

Documento	Documento original	Representación del documento mediante sus pares de palabras que GAP=1
D_1	$\langle a,b,f\rangle$	$\langle a,b\rangle_4\ \langle a,f\rangle_2\ \langle b,f\rangle_2$
D_2	$\langle b,c,d\rangle$	$\langle b,c\rangle_5\ \langle b,d\rangle_3\ \langle c,d\rangle_3$
D_3	$\langle a,b,c,d\rangle$	$\langle a,b\rangle_4\ \langle a,c\rangle_3\ \langle b,c\rangle_5\ \langle b,d\rangle_3\ \langle c,d\rangle_3$
D_4	$\langle a,b,c,d,e,a,b,f\rangle$	$\mathbf{\langle a,b\rangle_4}\ \langle a,c\rangle_3\ \mathbf{\langle b,c\rangle_5}\ \langle b,d\rangle_3\ \langle c,d\rangle_3\ \mathbf{\langle c,e\rangle_3}\ \langle d,e\rangle_1$ $\langle d,a\rangle_1\ \langle e,a\rangle_1\ \langle e,b\rangle_1\ \langle a,b\rangle_4\ \langle a,f\rangle_2\ \langle b,f\rangle_2$
D_5	$\langle b,c,e\rangle$	$\langle b,c\rangle_5\ \langle b,e\rangle_2\ \langle c,e\rangle_3$
D_6	$\langle a,b,c,e\rangle$	$\mathbf{\langle a,b\rangle_4}\ \langle a,c\rangle_3\ \mathbf{\langle b,c\rangle_5}\ \langle b,e\rangle_2\ \mathbf{\langle c,e\rangle_3}$

Siguiendo esta idea, Dimasp-C_n construye una estructura de datos a partir de la BDD en la cual es posible hacer la búsqueda de SFM's. Al igual que Dimasp-C_0, Dimasp-C_n mantiene una *Lista* Δ en la cual liga todas las ocurrencias de la secuencia $\langle a,b\rangle$ en la BDD, de manera que a partir de *Lista* Δ sea posible recuperar de manera rápida las posiciones donde ocurre la secuencia $\langle a,b\rangle$. Así, la frecuencia de la secuencia $\langle a,b\rangle$ en la colección estará determinada por el número de documentos diferentes que hay en *Lista* Δ. La figura 3.31 muestra una representación de la forma en que *Lista* Δ liga las ocurrencias de la secuencia $\langle a,b\rangle$ en la BDD.

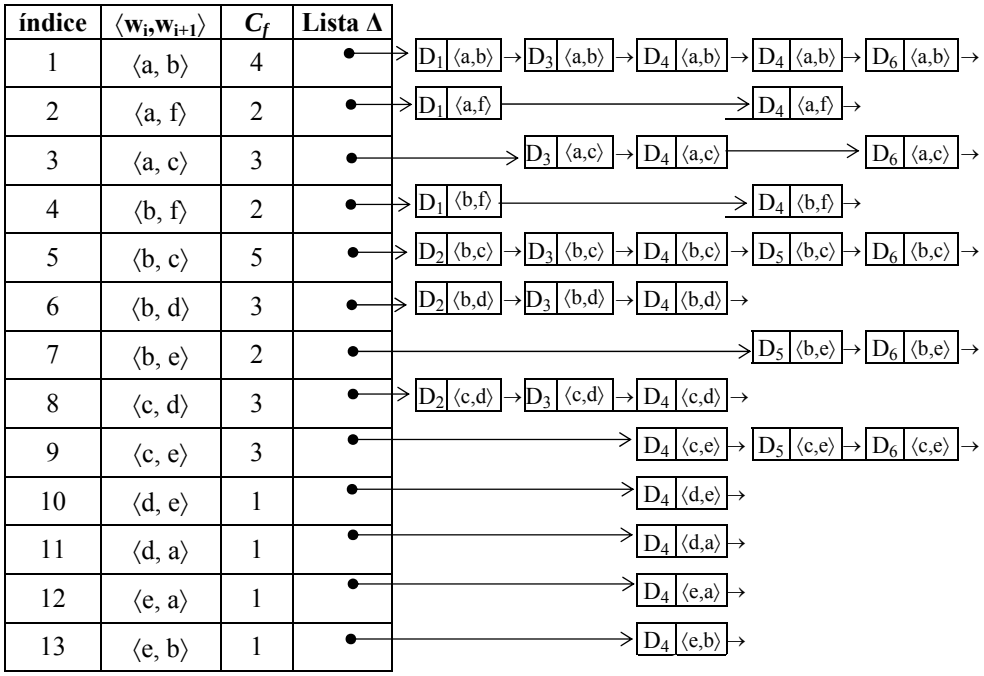


Figura 3.31 Arreglo de *Listas* Δ para los pares de palabras de la colección de documentos con GAP=1 utilizada en la tabla 3.3.

Como se observa en el arreglo de la figura 3.31, es posible obtener de cada *Lista* Δ el par de palabras, su frecuencia y la lista de ocurrencias de cada par en toda la BDD.

Hasta este momento se observa que la estructura de datos de Dimasp-C_n está basada en representar los documentos mediante los pares de palabras de cada documento de acuerdo a GAP, sin embargo los pares no preservan su estructura secuencial, por lo que todavía no es posible realizar el crecimiento de la secuencia frecuente. Esto puede resolverse creando un nuevo tipo de nodo por cada palabra del documento los cuales se ligan de acuerdo a la relación entre cada par de palabras que permite el GAP, lo que permite mantener el orden secuencial en el documento. La figura 3.32 nos muestra de manera simplificada la representación adoptada por Dimasp-C_n para un documento, la cual preserva la estructura secuencial en los pares de palabras del documento, para GAP=1.

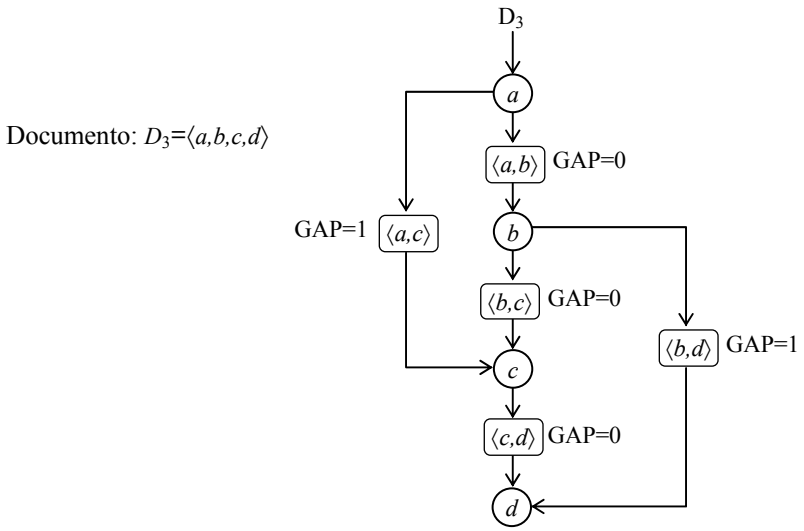


Figura 3.32 Representación adoptada por Dimasp-C_0 para el documento D_3 de la tabla 3.3.

La figura 3.33 muestra cómo son ligados los pares de palabras para los documentos de la figura 3.31.

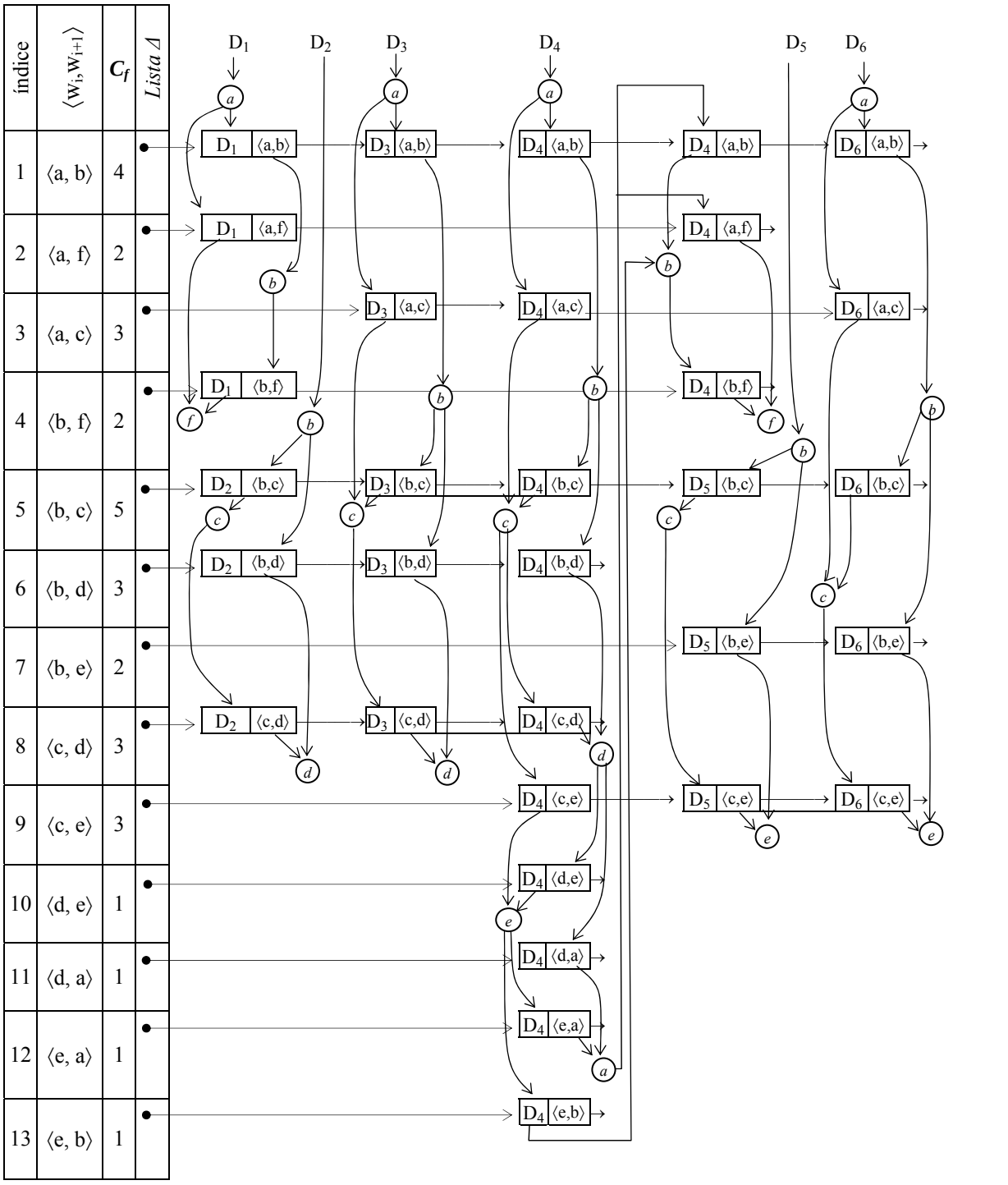


Figura 3.33 Estructura de datos construida en Dimasp-C_n en la cual están ligados los pares de palabras que el GAP=1 permite en la colección de documentos de la tabla 3.3.

Con el propósito de reducir la información almacenada en la figura 3.33, es posible sustituir los pares de palabras de las *Listas* Δ por el índice del par en el arreglo. También en la figura 3.33 es posible sustituir el identificador del documento por el número de documento analizado, lo cual se muestra en la figura 3.34.

También el espacio utilizado por la estructura de la figura 3.34 se puede reducir si en lugar de guardar el identificador del documento en el par de palabras se guarda en el nodo de cada palabra, de esta manera aunque el GAP se incremente no afectará en el espacio requerido para almacenar la estructura.

Con las características de la estructura de la figura 3.35 es posible hacer el descubrimiento de las SFM's, pues es posible recuperar todas las ocurrencias de un par de palabra en la BDD, sin perder el orden secuencial que guardan dentro del documento.

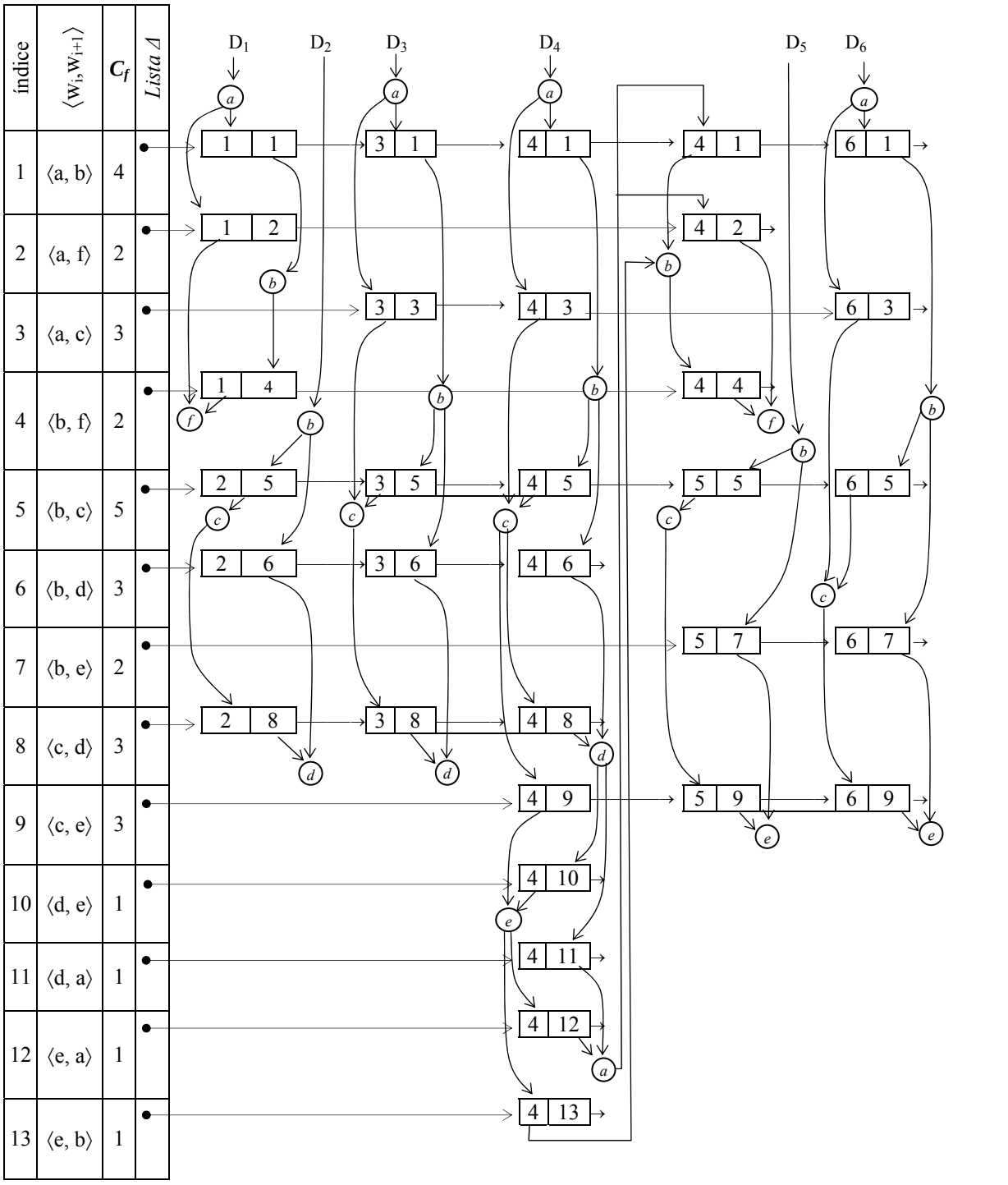


Figura 3.34 A partir de la estrucutra de datos de la figura 3.33 se sustituyeron los pares de palabras por su índice en el arreglo.

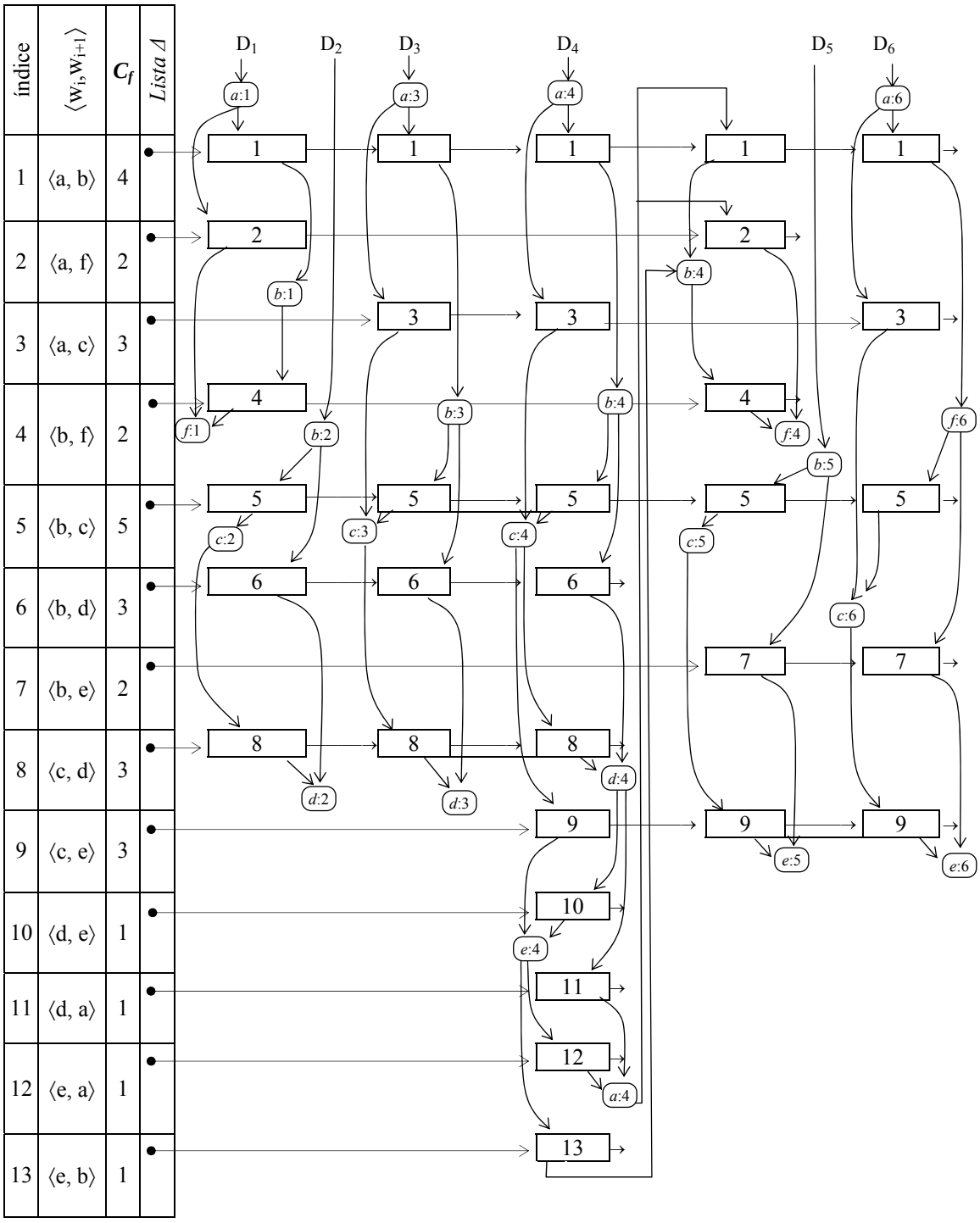


Figura 3.35 A paritr de la estructura de datos de la figura 3.34 se cambio de lugar el identificador del documento que estaba en el nodo δ por el nodo ω.

Como se puede apreciar, la idea básica de Dimasp-C_n consiste en encontrar todas las SFM's en la estructura de datos —construida a partir de la base de documentos (BDD)— la cual guarda todos los pares distintos de palabras de acuerdo al GAP que aparecen en los documentos, sin perder el orden secuencial. Dado un umbral β especificado por el usuario, Dimasp-C_n revisa si un par de palabras es frecuente, en cuyo caso se vuelve una secuencia frecuente (SF) de longitud 2 (2~SF). A partir de cada SF, Dimasp-C_0 crece, lo más posible, cada secuencia con el objetivo de determinar todas las SF's que contienen al par de palabras como prefijo. Una vez que una SF no puede crecer más entonces sólo falta determinar si esa SF no es subsecuencia de alguna otra SF, en cuyo caso es una SFM.

Al igual que Dimasp-C_0, Dimasp-C_n está dividido en 4 etapas. En la primera etapa se transforman y preparan los datos para la segunda etapa. En la segunda etapa se construye la estructura de datos a partir de la cual se realizará la búsqueda de secuencias frecuentes. En la tercera etapa, de acuerdo al umbral especificado por el usuario (β) se descubre para cada par de palabras la SF más larga que se pueda formar a partir de él en los documentos de la colección. Por último, en la cuarta etapa se seleccionan a partir del conjunto de SF's encontradas en la tercera etapa, las SFM's. La etapa de selección de SFM's para Dimasp-C_n se realiza de manera idéntica que en Dimasp-C_0 (sección 3.1.2.4).

3.2.2.1 Primera etapa, transformación de los documentos

Al igual que Dimasp-C_0, Dimasp-C_n asigna identificadores numéricos a cada una de las palabras y a cada par de palabras de la BDD. Dimasp-C_n además de asignar el identificador numérico al par de palabras $\langle w_{i-1},w_i\rangle$ de la BDD, también asigna identificadores numéricos a los pares de palabras que se pueden formar para el GAP especificado, correspondientes a $\langle w_{i-(GAP+1)},w_i\rangle$, $\langle w_{i-GAP},w_i\rangle$, …,$\langle w_{i-1},w_i\rangle$. Un ejemplo de los pares de palabras que se generan con GAP=1 para un documento se muestra en la figura 3.32.

Para lograr esto, se emplea la misma estructura de datos tipo árbol empleada en Dimasp-C_0, en la cual se desarrolla el mismo proceso para conocer el identificador numérico de cada palabra w_i de la BDD. Sin embargo, cuando en Dimasp-C_n se alcanza un nodo hoja, con el identificador numérico correspondiente a la palabra w_i, se toma ese nodo como la raíz del árbol a partir del cual se determinan los identificadores numéricos de los pares de palabras $\langle w_{i-(GAP+1)}, w_i\rangle$, $\langle w_{i-GAP}, w_i\rangle$, …, $\langle w_{i-1},w_i\rangle$. Este mismo procedimiento ya se llevaba a cabo con el par $\langle w_{i-1},w_i\rangle$ en Dimasp-C_0, pero ahora se generaliza el procedimiento para los GAP+1 pares. Para ello, a partir del nodo hoja actual, que actúa ahora como raíz del árbol, se agregan al

árbol los dígitos de los identificadores numéricos correspondientes a las palabras $w_{i-(GAP+1)}$, $w_{i-GAP},\ldots,w_{i-1}$. Por ejemplo, para agregar al árbol el identificador correspondiente a la palabra $w_{i-(GAP+1)}$, se avanza un nivel en el árbol de acuerdo a los dígitos de ese identificador, al final se recupera el *número* de ese nodo hoja como identificador del par de palabras $\langle w_{i-(GAP+1)}, w_i \rangle$.

La primera etapa de Dimasp-C_0 se realiza en el peor caso $O(n)$ tomando a n como el número palabras en la BDD. Esta relación en Dimasp-C_n se ve linealmente incrementada conforme crece el GAP, pues por cada palabra de la BDD ahora se consideran GAP+1 pares de palabras, por lo que en su peor caso la primera etapa se construye en tiempo $O((GAP+1)n)$

3.2.2.2 Segunda etapa, construcción de la estructura de datos

En la segunda etapa Dimasp-C_n utiliza casi las mismas estructuras de datos que Dimasp-C_0, esto es, el *arreglo* de pares de palabras y el nodo δ. La extensión que se hace en la construcción de la estructura de datos para permitir manejar un GAP=n reside básicamente en generar, en la estructura de datos, no solamente los pares contiguos de palabras de un documento sino en generar los distintos pares de palabras para cada GAP permitido. La figura 3.32 muestra los distintos pares de palabras de un documento para GAP=1. En el caso de Dimasp-C_0 se generaba, para cada palabra w_i de un documento, sólo el par de palabras $\langle w_{i-1}, w_i \rangle$, puesto que GAP=0. Sin embargo, en Dimasp-C_n se generan, por cada palabra w_i de un documento, los GAP+1 pares de palabras correspondientes a $\langle w_{i-(GAP+1)}, w_i \rangle$, $\langle w_{i-GAP}, w_i \rangle$, $\ldots,\langle w_{i-1}, w_i \rangle$. En Dimasp-$C_0$ cada par de palabras tiene asociado un nodo δ, el cual expresa la relación entre las dos palabras del par. Dimasp-C_n utiliza también un nodo δ para cada par de palabras, excepto que ya no se guarda el identificador del documento, quedando como se muestra en la figura 3.36. En este sentido, ahora en Dimasp-C_n se utiliza un nodo ω (ver figura 3.37) el cual contiene el identificador del documento $\omega.Id$, además de un arreglo con GAP+1 ligas a nodos δ mediante las cuales es posible asociar para cada palabra w_i, los GAP+1 pares de palabras que se pueden formar con ella. Otra diferencia es que la liga $\delta.NextPair$ ahora liga con un nodo ω.

Puede notarse que cuando el GAP crece el número de pares de palabras crece y por consiguiente el algoritmo utiliza más memoria, esto no constituye un problema ya que el valor del GAP en texto debe de ser pequeño.

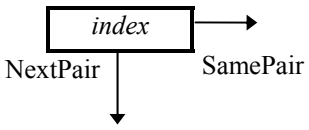


Figura 3.36 Estructura de un nodo δ para Dimasp-C_n.

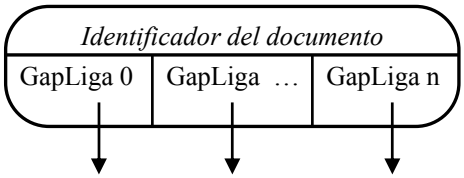


Figura 3.37 Estructura de un nodo ω para Dimasp-C_n.

La segunda etapa trabaja como sigue: para cada palabra w del documento D_j se crea un nodo ω el cual tiene a j como identificador del documento. De manera similar, para cada par de palabras $\langle w_{i-(GAP+1)}, w_i\rangle, \langle w_{i-GAP}, w_i\rangle, \ldots, \langle w_{i-1}, w_i\rangle$ del documento D_j se crean los nodo δ correspondientes a los pares de palabras $\delta(\langle w_{i-(GAP+1)}, w_i\rangle), \delta(\langle w_{i-GAP}, w_i\rangle), \ldots, \delta(\langle w_{i-1}, w_i\rangle)$. Después se asocian los nodos de manera que el nodo ω_p esté asociado (utilizando $\omega_p.GapLiga$) con el nodo $\delta(\langle w_p, w_q\rangle)$ y éste a su vez esté asociado (utilizando $\delta(\langle w_p, w_q\rangle).NextPair$) con el nodo ω_q. De esta manera, se mantiene la estructura planteada en la figura 3.32 para todas las palabras y pares de palabras.

Al igual que en Dimasp-C_0, los nodos δ son agregados a la *Lista* Δ respectiva (utilizando $\delta.SamePair$). La frecuencia de una celda se incrementa cuando se agrega un par del documento j y no se tiene registro de este documento en la *Lista* Δ. Todos los nodos δ de *Lista* Δ tienen asignado el mismo índice i de celda en $\delta.index=i$, de esta manera a partir de nodo δ es posible conocer tanto el par de palabras como su frecuencia. Es importante observar que los nodos δ tienen además otras funciones, por un lado, permiten mantener la *Lista* Δ, la cual registra todas las ocurrencias de un par de palabras en la BDD y, por otro lado, permiten mantener el orden secuencial que tienen los pares de palabras dentro de un documento. El algoritmo de la segunda etapa se muestra en la figura 3.38. La figura 3.39 muestra la estructura de datos construida en la segunda etapa usando la BDD de la tabla 3.3.

Si n es el número de palabras en la colección de documentos, se observa que para cada palabra w_i de la BDD se crean los nodos correspondientes a los pares de palabras $\delta(\langle w_{i-(GAP+1)}, w_i\rangle), \delta(\langle w_{i-GAP}, w_i\rangle), \ldots, \delta(\langle w_{i-1}, w_i\rangle)$ en la colección de documentos. Si m es el número de pares de palabras diferentes en BDD entonces el arreglo de pares de palabras es de tamaño

m. Por lo que el algoritmo de la segunda etapa requiere un espacio $\Theta(n(GAP+1)+m)$ donde $m<(GAP+1)n$ y un tiempo $\Theta(n(GAP+1))$.

Etapa 2: Algoritmo para construir la estructura de datos a partir de BDD

Entrada: Una base de datos de documentos (BDD) transformada en la etapa 1

Salida: La estructura de datos de la BDD

Para *todos los documentos* $D_j \in BDD$ **hacer**

    Arreglo ← Agregar el documento (D_j) al arreglo

fin-para

Etapa 2.1: Algoritmo para agregar un documento a la estructura de datos

Entrada: Un documento D_j; BDD transformada; **Salida:** Estructura de datos

Para *todos los pares de palabras* $\langle w_i, w_{i+1}\rangle \in D_j$ **hacer**

    $\omega_i.Id \leftarrow j$ *//Asignar el identificador del documento al nodo* ω_i

    Para $gap=1...GAP+1$ **hacer**

        $\delta_i(\langle w_i, w_{i+gap}\rangle) \leftarrow$ *Crear el nuevo nodo* δ *correspondiente a* $\langle w_i, w_{i+gap}\rangle$

        índice ← $Arreglo[\langle w_i, w_{i+gap}\rangle]$ *//Obtener el índice de la celda de* $\langle w_i, w_{i+gap}\rangle$, *calculado en la etapa 1*

        $\delta_i(\langle w_i, w_{i+gap}\rangle)$ *.index* ← *índice* *//Asignar el índice al nodo* δ

        $\alpha \leftarrow Arreglo[índice].Lista\ \Delta$ *//Obtener el primer nodo de la lista* Δ

        Si $j \neq \alpha.\omega_i.Id$ **entonces** *identificador del documento es nuevo en Lista* Δ

            $Arreglo[índice].Lista\ \Delta.C_f$ ++ *//incrementar la frecuencia de la celda*

        $Arreglo[índice].Lista\ \Delta \leftarrow \delta_i.SamePair \leftarrow \alpha$ *//Ligar el nodo al comienzo de Lista* Δ

        Establecer la asociación $\omega_i \rightarrow \delta_i(\langle w_i, w_{i+gap}\rangle) \rightarrow \omega_{i+gap}$ *esto se hace con las siguientes 2 líneas:*

            $\delta_i(\langle w_i, w_{i+gap}\rangle).NextPair \leftarrow \omega_{i+gap}$

            $\omega_i.GapLiga[gap-1] \leftarrow \delta_i(\langle w_i, w_{i+gap}\rangle)$

    fin-para

fin-para

Figura 3.38 Algoritmos para la etapa 2 y 2.1 de Dimasp-C_n donde se construye la estructura de datos para los documentos de la colección.

índice	⟨w_x,w_y⟩	C_f	¿el par fue usado?	Lista Δ
1	⟨The,first⟩	1	No	•
2	⟨The,President⟩	2	No	•
3	⟨first,President⟩	1	No	•
4	⟨The,of⟩	1	No	•
5	⟨first,of⟩	1	No	•
6	⟨President, of⟩	2	No	•
7	⟨President,the⟩	1	No	•
8	⟨President,United⟩	1	No	•
9	⟨of,the⟩	2	No	•
10	⟨of,United⟩	2	No	•
11	⟨the,United⟩	2	No	•
12	⟨the,States⟩	2	No	•
13	⟨United,States⟩	2	No	•
14	⟨United,was⟩	2	No	•
15	⟨States,was⟩	1	No	•
16	⟨States,is⟩	1	No	•
17	⟨States,George⟩	2	No	•
18	⟨is,George⟩	1	No	•
19	⟨was,George⟩	1	No	•
20	⟨was,Washington⟩	1	No	•
21	⟨is,Bush⟩	1	No	•
22	⟨George,Washington⟩	1	No	•
23	⟨George,Bush⟩	1	No	•

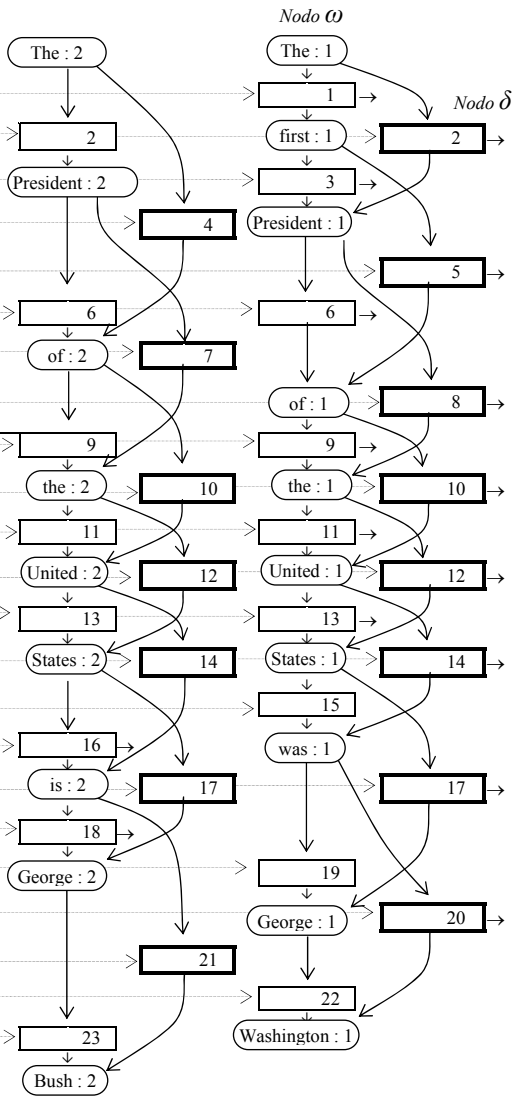


Figura 3.39 Estructura de datos construida con GAP=1 para la colección de documentos compuesta por el documento "*The first President of the United States was George Washington"* y el documento *"The President of the United States is George Bush"*. Nota, los nodos ω son representados por un óvalo el cual contiene a la palabra w_i que representa. Los nodos δ son representados por un rectángulo con línea sencilla para aquellos creados cuando GAP=0 y con línea más gruesa para aquellos creados cuando GAP=1.

3.2.2.3 Tercera etapa, búsqueda de SF's de acuerdo al GAP y al umbral β

En el algoritmo Dimasp-C_0, el crecimiento de una secuencia frecuente se hace siempre con el siguiente par de palabras, siguiendo de esta manera un sólo camino en la construcción de la SF, pues el GAP=0 sólo así lo permitía. Sin embargo, en Dimasp-C_n el crecimiento de una secuencia frecuente puede seguir $n+1$ caminos en cada crecimiento pues GAP=n así lo permite, por lo que se podría producir más de una SF. Por tal motivo se hizo necesario controlar el crecimiento de las secuencias frecuentes (SF's) con una estructura de datos tipo árbol. Con el objetivo de esclarecer cómo el árbol es usado para desarrollar la búsqueda a continuación se muestra el análisis de un ejemplo.

Suponiendo que se tiene una colección de m documentos idénticos donde $m>2$ y el umbral requerido es $\beta=2$, por ejemplo:

$Doc\ 1=\langle A,B,C,D,E,F,G,H\rangle$
$Doc\ 2=\langle A,B,C,D,E,F,G,H\rangle$
$Doc\ldots=\langle A,B,C,D,E,F,G,H\rangle$
$Doc\ m=\langle A,B,C,D,E,F,G,H\rangle$

Para Dimasp-C_0 y con la colección anterior, el crecimiento de una SF sólo podía hacerse en un sólo camino o rama, produciendo una sola SF. Sin embargo, en Dimasp-C_n el crecimiento de la SF se haría en varias ramas es decir en forma de árbol. Por ejemplo, utilizando la colección de secuencias anterior y con $GAP=1$ se producirían tantos caminos como los que se muestran en el árbol de la figura 3.40 en donde cada una de las ramas en el árbol desde la raíz hasta las hojas conforman una SF, en este caso se producen 21 SF's.

Como se observa, para hacer el crecimiento de las SF's de la figura 3.40 se tienen que procesar y guardar en cada nodo del árbol el estado del crecimiento de las SF's, es decir, se tiene que guardar el registro de las palabras de los m documentos que hacen a esa secuencia frecuente. Aunque el ejemplo anterior es un caso especial de secuencias, es común que los algoritmos de minería de patrones secuenciales tomen un tiempo considerable para procesar colecciones similares con documentos más largos. Debido a que el proceso de enumeración de las SFM's es exponencial, en la etapa del crecimiento de las SF's sería conveniente evitar el procesamiento de subárboles iguales. Para esto es necesario

identificar los nodos que procesan a las *mismas palabras*[2] de acuerdo a cada documento, ya que estos nodos producen el mismo subárbol. De esta manera, los nodos redundantes sólo se procesan una sola vez. La figura 3.41 muestra la conexión de nodos que se hace al identificarse nodos redundantes, los subárboles no conectados en la figura 3.41 tienen como objetivo ejemplificar la cantidad de trabajo evitado al hacer el crecimiento de SF's mediante la identificación de nodos redundantes.

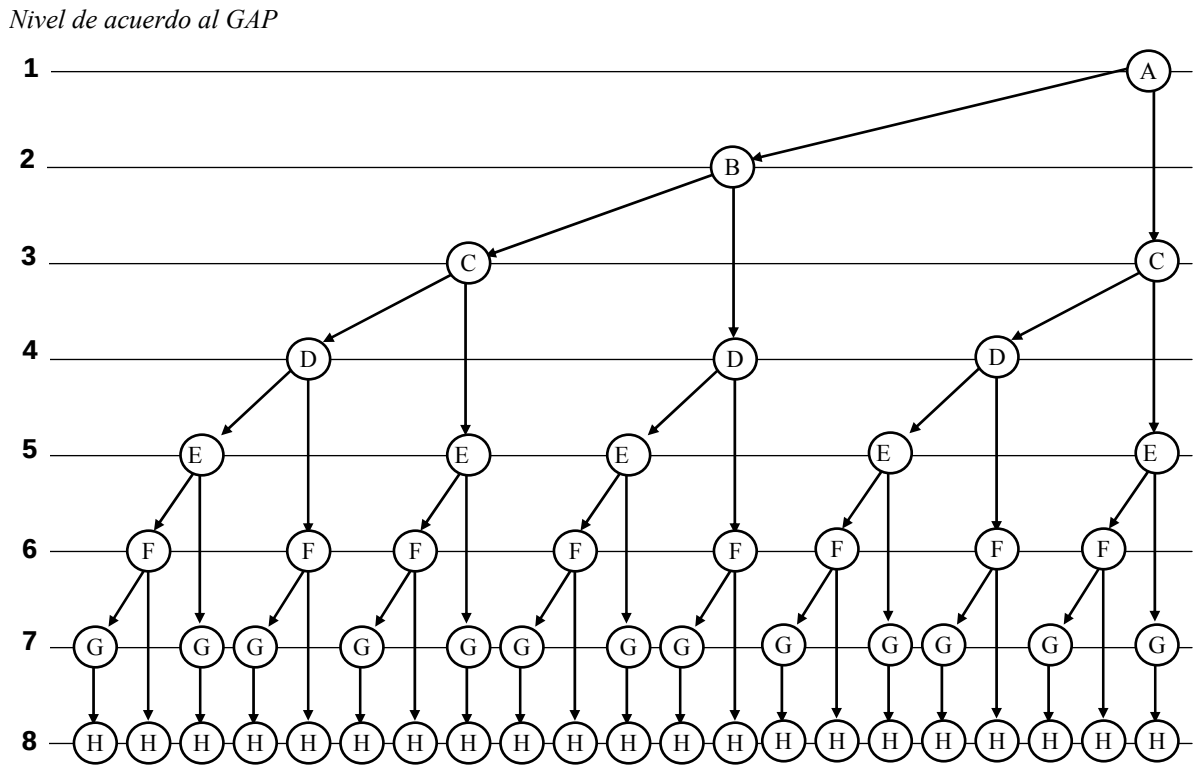


Figura 3.40 Árbol de búsqueda de SF's desarrollado por Dimasp-C_n para *m* documentos idénticos $\langle A,B,C,D,E,F,G,H \rangle$.

La identificación de subárboles es una estrategia para el crecimiento de SF's que, por un lado, acelera la búsqueda y, por otro lado, elimina la necesidad de almacenar, para cada nodo, una gran cantidad de información relativa al estado del crecimiento de la SF para ese nodo. Sin embargo, debido a la gran cantidad de nodos generados esta estrategia sigue

[2] "la misma palabra" se refiere a que la palabra esté en el mismo nivel del árbol de acuerdo al GAP y en los mismos documentos, ver figura 3.40.

siendo ineficiente en espacio. Para solucionar este problema se diseñó otra estrategia que permite ir procesando y liberando de manera controlada los nodos del árbol. Como es sabido, el procesamiento de un árbol se puede hacer en profundidad o en amplitud. No se hace en profundidad porque se tendría que guardar el estado de todos los nodos hasta llegar a producir la SF más larga. Sin embargo, si procesamos el árbol en amplitud, pero de acuerdo al nivel del GAP, se tiene la posibilidad de liberar la memoria utilizada por los nodos de menor nivel en el árbol, puesto que esos nodos no pueden volver a ser procesados. Además la estrategia en amplitud no sólo nos permitiría liberar la memoria de los nodos del nivel menor, sino que nos permitirá reutilizar esa memoria en los nodos del nivel mayor.

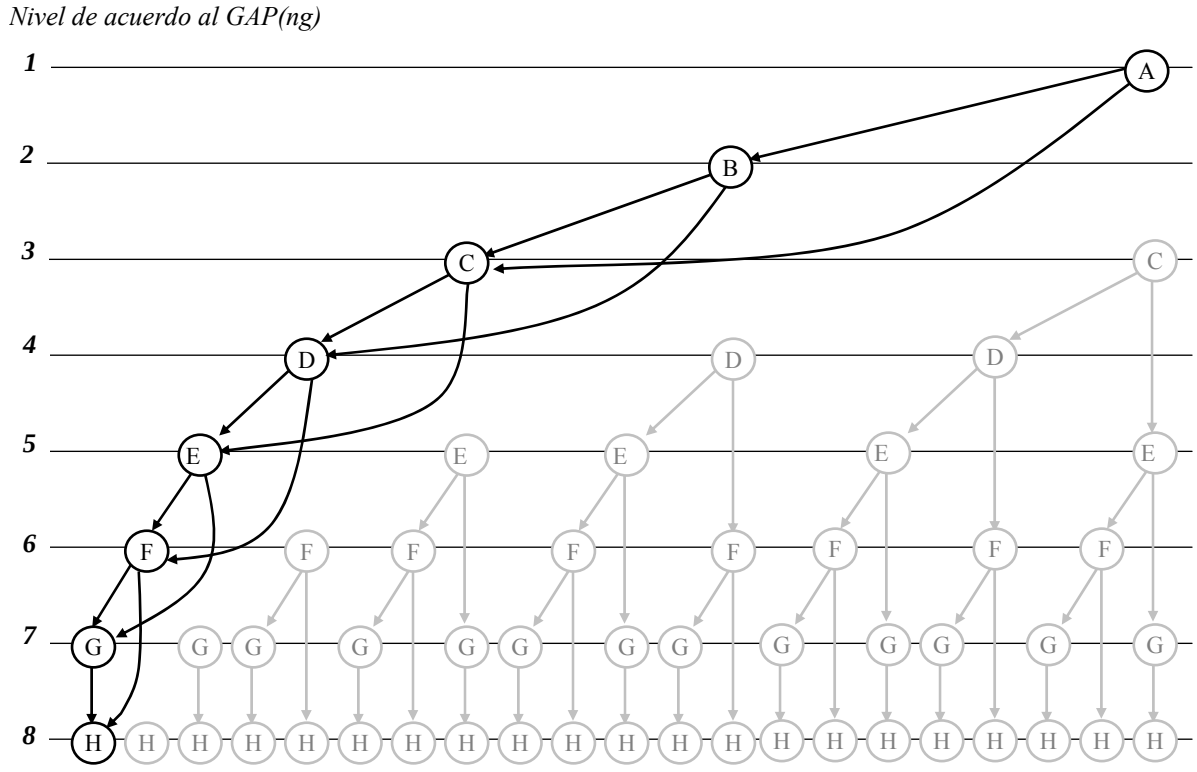


Figura 3.41 Ejemplo del árbol de crecimiento de las SF's desarrollado en la tercera etapa por Dimasp-C_n para *m* documentos idénticos $\langle A,B,C,D,E,F,G,H \rangle$.

Por ejemplo, en la figura 3.40 se observa que al analizar el nivel 1 correspondiente al nodo '*A*' se generan los niveles 2 y 3; y para el nivel 1 no existe posibilidad de volver a generar un nodo en ese nivel, por lo que esa memoria puede ser liberada y asignada al nivel 4. Luego se procesaría el nivel 2 correspondiente a la palabra 'B' el cual genera los niveles de nodos 3 y

4; y al finalizar todos los nodos de este nivel, la memoria del nivel 2 puede ser liberada y asignada al nivel 5. Luego se procesaría el nivel 3 correspondiente a la palabra 'C' el cual genera dos niveles de nodos 4 y 5; y, al finalizar todos los nodos de este nivel, la memoria del nivel 3 puede ser liberada y asignada al nivel 6. Generalizando, cuando se procesan los nodos del nivel-gap ng se producen $ng+1, ng+2, \ldots, ng+(GAP+1)$ niveles y una vez procesados todos los nodos del nivel ng se puede reasignar la memoria del nivel ng al nivel $ng+GAP+2$.

Una vez construida la estructura de datos en la etapa 2, en la tercera etapa Dimasp-C_n hace uso de la estrategia de crecimiento por amplitud y de la identificación de subárboles idénticos para encontrar todas las SFM's de acuerdo al umbral β y GAP especificado por el usuario. Para lograr esto, Dimasp-C_n hace lo siguiente: para cada celda del arreglo con una frecuencia mayor o igual a β se hace el crecimiento por amplitud de cada uno de los nodos δ contenidos de la *Lista* Δ, esto con el objetivo de construir el árbol que determina las SF's que empiezan con el par correspondiente a dicha celda. Para construir el árbol se utilizan nodos φ, los cuales contienen información acerca del estado de crecimiento de las SF's, esto es, la palabra w relativa a la SF, el *nivel* del nodo de acuerdo al GAP, una *Lista* Ω que contiene ligas a nodos ω, la *frecuencia* que está determinada por el número de documentos diferentes que aparecen en la *Lista* Ω, además de un arreglo de *GapHijos* con GAP+1 ligas a otros nodos φ.

Para hacer el crecimiento de un nodo $\delta(\langle w_x, w_y \rangle)$ se crea el nodo $\varphi(\langle w_x, w_y \rangle)$ correspondiente a la raíz del árbol el cual incluye como palabra $\varphi.w$ a la primera palabra (w_x) de la celda $\delta.index$, '1' como $\varphi.nivel$ del nodo y la *Lista* Ω está conformada por todos los nodos asociados ω_y que se tienen en $\delta(\langle w_x, w_y \rangle) \rightarrow \omega_y$. Cabe recordar que la *frecuencia* del nodo φ está calculada con base en el número de documentos diferentes en *Lista* Ω. Una vez incluido el nodo raíz φ como primer nivel del árbol se hace el crecimiento de las SF's; procesando todos los nodos φ de cada uno de los niveles en orden ascendente. El procesamiento de un nodo φ tiene como objetivo crear sus GAP+1 hijos posibles los cuales son agregados en el árbol si son frecuentes y si no existe otro nodo idéntico en el mismo nivel-gap.

A partir de un nodo $\varphi(\langle w_x, w_y \rangle)$ se puede crear un hijo $\varphi_g(\langle w_y, w_z \rangle)$ donde $0 \leq g \leq GAP+1$. Para construir el nodo φ_g se asigna w_y como la palabra $\varphi_g.w$, como $\varphi_g.nivel$ a $\varphi.nivel+g+1$ y en $\varphi_g.$*Lista* Ω se agregan los nodos ω_z de la lista $\varphi.$*Lista* Ω que cumplan la asociación $\omega_y \rightarrow \delta(\langle w_y, w_z \rangle) \rightarrow \omega_z$. Si al finalizar de procesar el nodo φ_g se tiene una frecuencia mayor o igual que β entonces se agrega φ_g al árbol en el nivel $\varphi_g.nivel$. Siempre que se agrega un nodo a un nivel se verifica si existe otro nodo idéntico a él.

Una vez procesados todos los nodos de un nivel entonces pueden ser eliminados y se continúa con el procesamiento de otro nivel. Al finalizar la construcción del árbol se generan las SF's recorriendo cada una de las ramas del árbol desde la raíz hasta las hojas. Una k–SF solamente se agrega al conjunto de SF's si $k \geq 3$, en tal caso, todas las celdas del arreglo usadas en la SF son marcadas como "usadas". Después se utiliza el procedimiento de la etapa 3.2 en la figura 3.19 del algoritmo de Dimasp-C_0 para determinar las SFM's de longitud 1 y 2. De igual forma, una vez procesadas todas las celdas del arreglo de pares de palabras se utiliza el procedimiento de la etapa 4 de Dimasp-C_0 (sección 3.1.2.4) para determinar cuáles de las SF's generadas son maximales, encontrando así las SFM's finales. En la figura 3.42 se muestra el algoritmo de la etapa 3.

Etapa 3: Algoritmo para encontrar todas las SF's
Entrada: Estructura de la etapa 2, el umbral β y el GAP **Salida:** Conjunto de SF's
$\varphi \leftarrow$ Nodo del Árbol, que contiene (*palabra* w, nivel, Lista Ω *de nodos* $\omega_{1..t}$)
Para *todos los índices de las celdas* $\in$ *Arreglo* **hacer**
  Si *Arreglo*[*índice*].$C_f \geq \beta$ **entonces** *la celda es frecuente, hacer el crecimiento de las SF's*
    Para *todos los nodos* $\delta_i \in$ *Arreglo*[*índice*].*Lista* Δ **hacer**
      $\varphi \leftarrow$ *Crear* (*Arreglo*[*índice*].w_i , 1, { $\delta \in$ *Arreglo*[*índice*].*Lista* Δ | $\delta_i \rightarrow \omega_x$ }/ agregar a ω_x y no a δ_i)
      Árbol.Raíz $\leftarrow \varphi$
      Árbol.GapNivel(1) $\leftarrow$ *Árbol.Raíz*
      Para *todos los niveles-gap ng del Árbol* **hacer** *el procesamiento por amplitud*
        Para *todos los nodos* φ *del nivel ng* **hacer** *el crecimiento de nodos*
          Para $\varphi.\omega_1 \rightarrow \delta(\langle w_1, w_g \rangle) \rightarrow \omega_g$ *donde* $g=1...GAP+1$ **hacer** *el crecimiento para sus GAP+1 hijos*
            $\varphi_g \leftarrow$ *Crear nodo* (w_1 , φ.*Nivel*+g+1, { $\omega_g \in \varphi$.Lista Ω | $\omega_1 \rightarrow \delta(\langle w_1, w_g \rangle) \rightarrow \omega_g$}) *//crecer hijo*
            Si φ_g.*frecuencia* $\geq \beta$ **entonces** *agregar al nivel correspondiente*
              Arreglo [$\langle w_{i+(k-1)}, w_{i+(k)} \rangle$].*marca* $\leftarrow$ "usada"
              φ.*GapHijos*[g] $\leftarrow$ Buscar nodo redundante (φ_g)
              Árbol.GapNivel(φ.*nivel*+g+1) $\leftarrow \varphi_g$
            fin-si
          fin-para
        fin-para
      fin-para
      SF's $\leftarrow$ *Cada rama desde la raíz del Árbol a las hojas es una SF, las palabras de la SF son en* $\varphi.w$
      Árbol.GapNivel(ng+GAP+2) $\leftarrow$ Reutilizar memoria *Árbol.GapNivel(ng)*
    fin-para
  fin-si
fin-para

Figura 3.42 Algoritmo para encontrar todas las SF's usando la estructura de datos construida en la segunda etapa y un umbral β.

El peor caso de la tercera etapa sucede cuando se produce el mayor número de secuencias de la mayor longitud posible. Este caso en particular sucede cuando la colección está formada por n documentos idénticos con m palabras diferentes y con el umbral más

bajo $\beta = 2$. En tal caso, todas las m celdas del arreglo son frecuentes y para cada celda el crecimiento de la SF utilizaría los n documentos de la colección. En la figura 3.40 se puede apreciar que cuando m vale 1 entonces el número de nodos generados en el árbol $T(m)$ vale 1 (esto se puede ver porque hasta el nivel-gap=1 sólo hay un nodo) y si m=2 $T(m)$=2 (esto se puede ver porque hasta el nivel-gap=2 hay 2 nodos), si m=3 $T(m)$=4 (esto se puede ver porque hasta el nivel-gap 3 hay 4 nodos), si m=4 $T(m)$=7 (esto se puede ver porque hasta el nivel-gap 4 hay 7 nodos), es decir, generalizando para GAP=1, $T(m)=T(m-1)+T(m-2)+1$, lo cual es igual a:

$$T(m) = \left(\frac{1}{\sqrt{5}}\right)\left[\left(\frac{1+\sqrt{5}}{2}\right)^m - \left(\frac{1-\sqrt{5}}{2}\right)^m\right]$$

Además el crecimiento del árbol debe hacerse n veces por lo que para GAP=1 el orden del algoritmo es:

$$O(T(m)) = O\left[\left(\frac{1+\sqrt{5}}{2}\right)^m\right]$$

Generalizando de acuerdo al GAP se plantea la recurrencia homogénea:

$$T(m)=T(m-1)+T(m-2)+\ldots+T(m-(GAP+1))+1$$

Por lo que su ecuación característica para recurrencias homogéneas sería:

$$T(m) = C_1 r_1^m + C_2 r_2^m + \cdots + C_{GAP+1} r_{GAP+1}^m$$

Siempre y cuando el polinomio tenga m raíces diferentes. Lo cual nos dice que el orden del algoritmo es exponencial, lo que varía en la expresión es la base de acuerdo al GAP analizado.

3.2.3 Experimentación

De manera similar a Dimasp-C_0, los experimentos se llevaron a cabo usando la colección de documentos Reuters-21578 [LEWIS] la cual contiene 21578 documentos. Después de eliminar 226 ***stop-words*** (ver anexo 1), la colección tiene alrededor de 39,688 palabras diferentes de 1.18 millones de palabras usadas en toda la colección. Después de eliminar las ***stop-words*** la

longitud promedio de los documentos es de 55 palabras. En todos los experimentos fueron usados los primeros 5000, 10000, 15000 y 20000 documentos de la colección Reuters-21578. En la tabla 3.7 se muestran algunas de las características de las colecciones utilizadas.

Dimasp-C_n fue programado en ***Builder C++ versión 6.*** Con el objetivo de permitir trabajar con colecciones de documentos más grandes se decidió que Dimasp-C_n guardará los resultados de la etapa 1, 3 y 4 en archivos en disco duro los cuales son utilizados por su siguiente etapa. Por tal motivo en algunas de las siguientes gráficas se presenta el tiempo que toman los experimentos con y sin escritura de las etapas 3 y 4. Los experimentos aquí mostrados fueron ejecutados sobre una computadora Pentium 4 a 3Ghz con 1GB de memoria principal (RAM).

En la figura 3.43 se muestra la gráfica del tiempo de procesamiento para cada etapa de Dimasp-C_n con GAP=0 y β=15; sin escritura de archivos. En este caso la figura 3.43 muestra que el mayor tiempo es tomado por la primera etapa.

Tabla 3.7 Características de los subconjuntos de documentos de la colección Reuters-21578, las cuales son utilizadas para experimentación.

Número de documentos	**5,000**	**10,000**	**15,000**	**20,000**
Tamaño en MB	2.1	4.3	6.3	8.6
Caracteres procesados	1,854,971	3,835,773	5,558,871	7,554,850
Número de palabras	275,579	568,565	828,174	1,124,898
Palabras diferentes	19,962	27,874	33,115	38,488
Pares de palabras diferentes con GAP=0	193,771	367,393	505,937	661,599
Pares de palabras diferentes con GAP=1	391,225	740,002	1,016,773	1,325,608
Pares de palabras diferentes con GAP=2	577,224	1,087,596	1,488,430	1,934,657
Pares de palabras diferentes con GAP=3	750,693	1,408,042	1,920,649	2,489,438

Utilizando el mismo umbral β=15 se hizo variar el GAP desde 0 hasta 3 para procesar las diferentes particiones de la colección Reuters-21578, en la gráfica de la figura 3.44 se muestran los tiempos requeridos sin escritura de archivos y en la figura 3.45 con escritura de archivos. Con el objetivo de apreciar mejor las figuras 3.44 y 3.45 se omitieron los tiempos con GAP 2 y 3 para 20,000 documentos. En la figura 3.45 se muestra sin escritura de archivos, para GAP=2 donde se requirieron 197 segundos y para GAP=3 se requirieron 728 segundos. En cambio, con escritura de archivos, figura 3.45, para GAP=2 requirieron 284 segundos y para GAP=3 se requirieron 883 segundos.

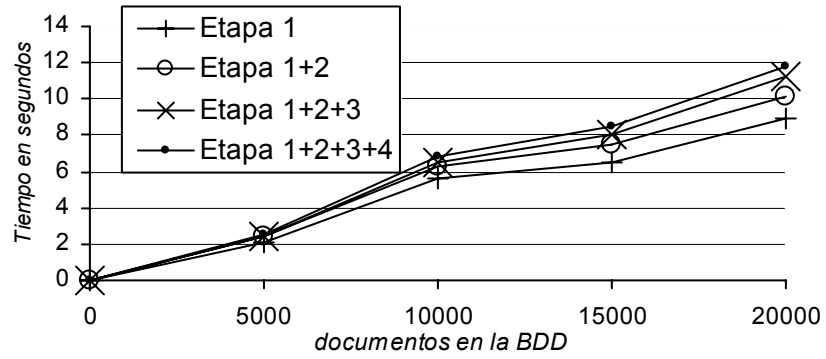


Figura 3.43 Tiempos de procesamiento para cada una de las etapas con Dimasp-C_n con GAP = 0 y β = 15; sin escritura de archivos.

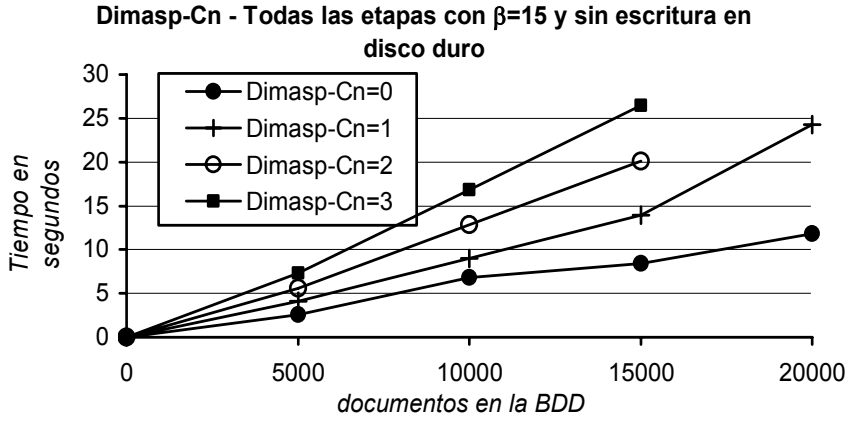


Figura 3.44 Desempeño de Dimasp-C_n con β = 15 y variando GAP de 0 a 3, sin escritura en disco duro.

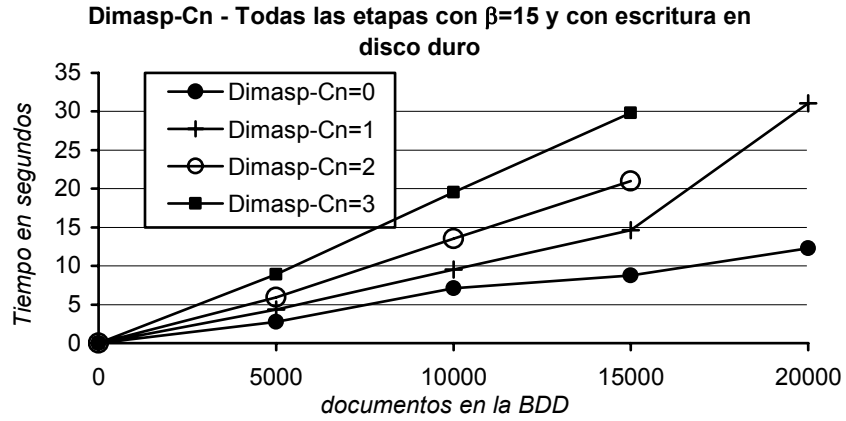


Figura 3.45 Desempeño de Dimasp-C_n con β = 15 y variando GAP de 0 a 3, con escritura en disco duro.

Con el objetivo de mostrar el desempeño de Dimasp-C_n frente a otros algoritmos se comparó el tiempo requerido por Dimasp-C_n contra los tiempos de cSPADE, GenPrefixSpan, DELISP y GSP. Para estos experimentos se utilizaron los programas originales suministrados por los autores de los mismos, excepto para GSP, el cual fue programado por nosotros. Para el experimento de la gráfica 3.47 se hicieron las pruebas estableciendo en los programas que "no se escribieran archivos". En la figura 3.46 se muestra la comparación de los algoritmos Dimasp-C_n, cSPADE, GenPrefixSpan, DELISP y GSP usando β=15 y GAP=1.

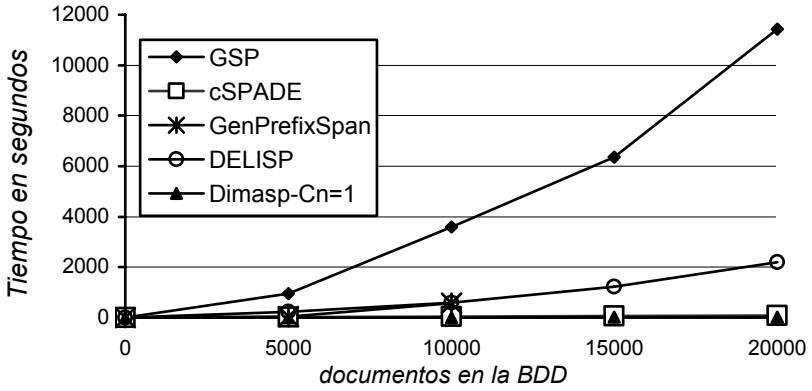


Figura 3.46 Comparación de Dimasp-C_n con otros algoritmos usando β = 15 y GAP = 1, sin escritura de archivos.

La figura 3.47 muestra la misma comparación de la figura 3.46 pero ahora se utilizó GAP=2.

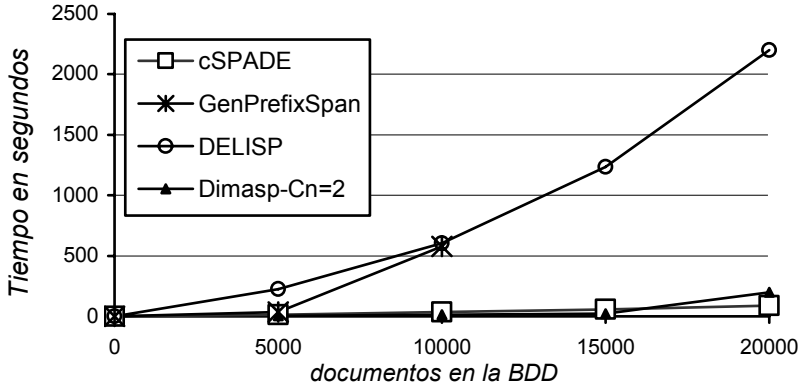


Figura 3.47 Comparación de Dimasp-C_n con otros algoritmos usando β = 15 y GAP = 2, sin escritura de archivos.

La figura 3.48 muestra la comparación del desempeño de cSPADE, Dimasp-C_0 y Dimasp-C_n con β=15 y GAP=0. En esta gráfica es posible apreciar como Dimasp-C_0 es ligeramente más rápido que Dimasp-C_n, para GAP=0.

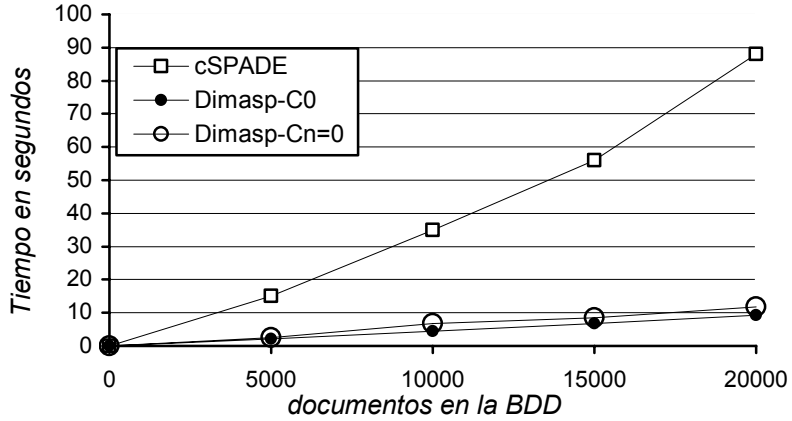


Figura 3.48 Comparación entre Dimasp-C_0 y Dimasp-C_n con $\beta = 15$ y GAP = 0, sin escritura de archivos.

El conjunto de gráficas mostradas en la figura 3.49 corresponden a la distribución de las SFM's según su longitud en cada experimento. En esta figura se puede apreciar que cuando GAP=0 la mayor cantidad de SFM's está concentrada en las de menor longitud 1 y 2. Sin embargo, al aumentar el GAP aumenta la cantidad de SFM's las cuales tienen una longitud entre 11 y 22. Esto explica en parte el aumento del tiempo de procesamiento para los experimentos de la figura 3.44 y 3.45 con 20000 documentos.

En los objetivos de esta tesis se especifica que el GAP en texto debe tener valores pequeños porque mientras mayor sea el GAP mayor la perdida del contexto de las palabras que forman la SFM. Aunado a esto, a partir de la gráfica de la distribución de SFM's con GAP=3 en la figura 3.49, se fortalece la idea que para GAP mayores a 3 (incluso a 2) encontrar SFM's en texto empieza a ser impráctico, debido a la gran cantidad de SFM's que se pueden generar.

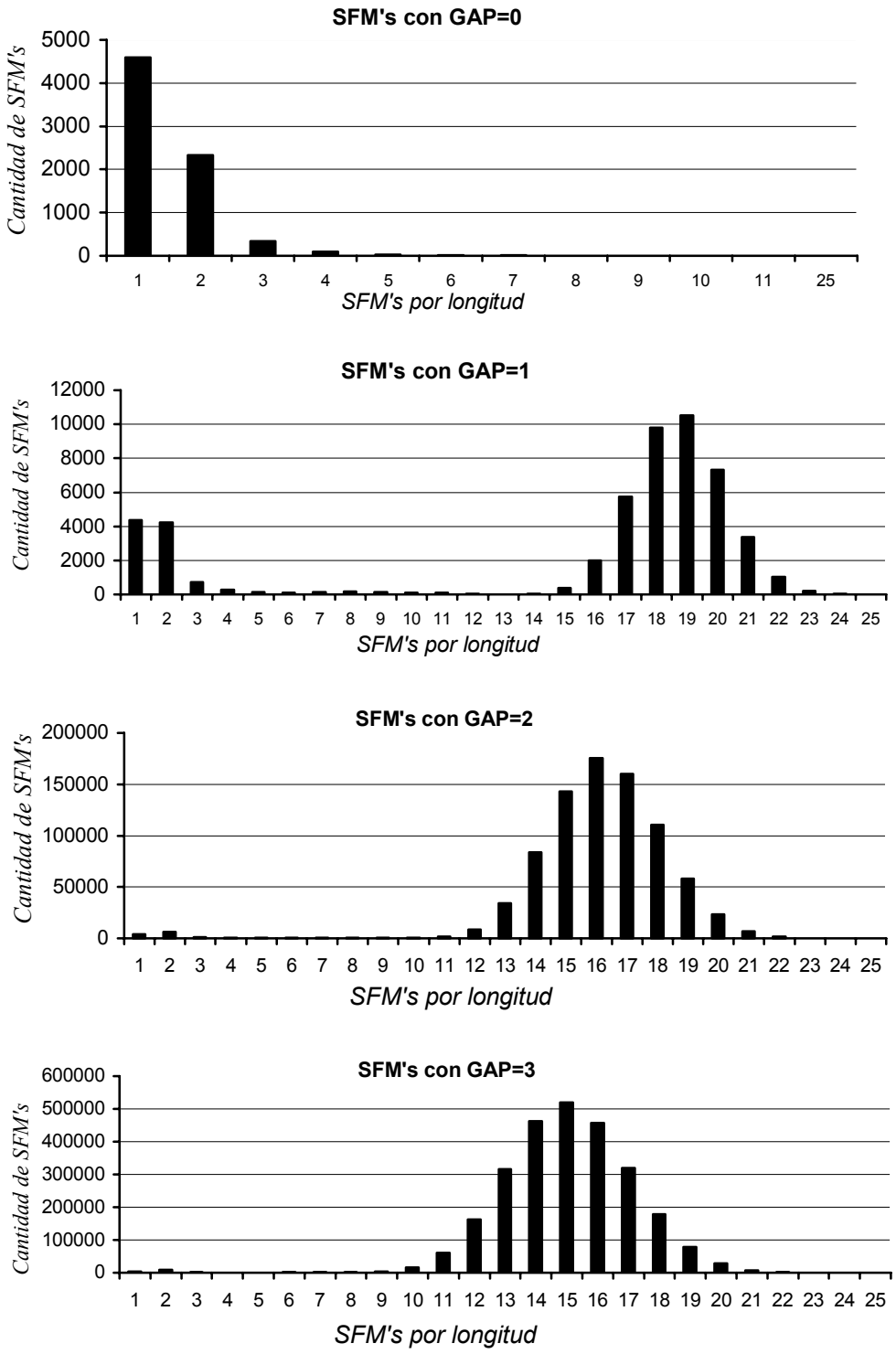


Figura 3.49 Distribución de las k~SFM's para 20000 documentos con $\beta = 15$ y GAP entre 0 y 3.

Con el propósito de apreciar cómo se afecta el desempeño de Dimasp-C_n en cada una de sus etapas de acuerdo al GAP se incluyen a continuación los experimentos utilizando β=15, pero variando el GAP entre 0 y 3. La gráfica 3.51 muestra la comparación para la primera etapa. En esta gráfica se observa que el tiempo requerido por la etapa 1 presenta un comportamiento lineal cuando se incrementa el número de documentos.

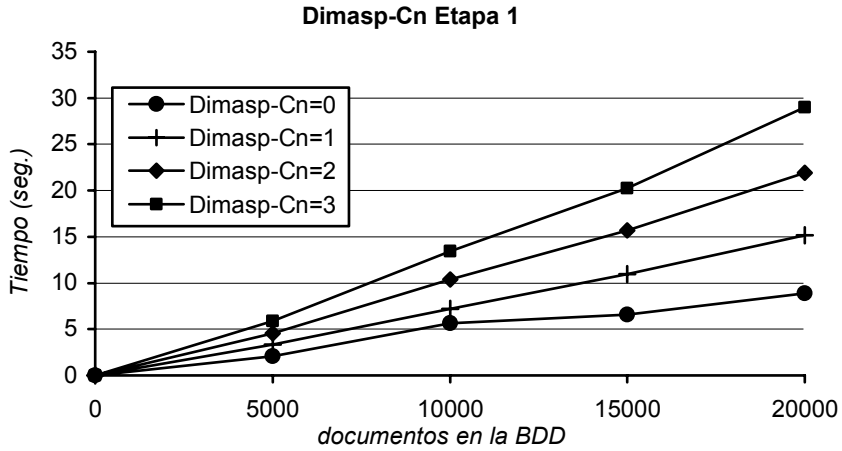


Figura 3.50 Tiempos obtenidos para la primera etapa de Dimasp-C_n con β = 15, para GAP=0,1,2,3.

La figura 3.51 muestra los tiempos requeridos para la construcción de la estructura de datos variando el GAP entre 0 y 3, sin escritura de archivos.

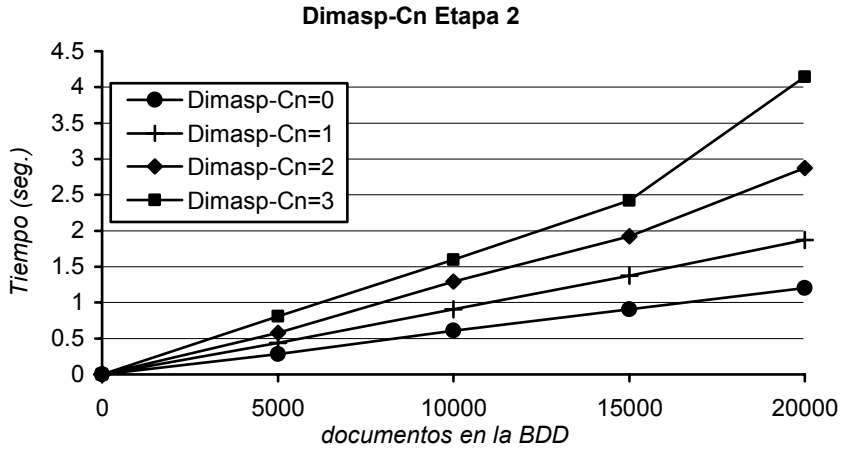


Figura 3.51 Tiempos obtenidos para la segunda etapa de Dimasp-C_n, para GAP = 0, 1, 2, 3.

En la figura 3.52 se presentan los tiempos requeridos por la tercera etapa con y sin escritura de archivos temporales. Para efectos de una mejor visualización se omitió en la figura 3.52(b) los 265 segundos requeridos para procesar la etapa 3 con GAP 3 para 20000 documentos.

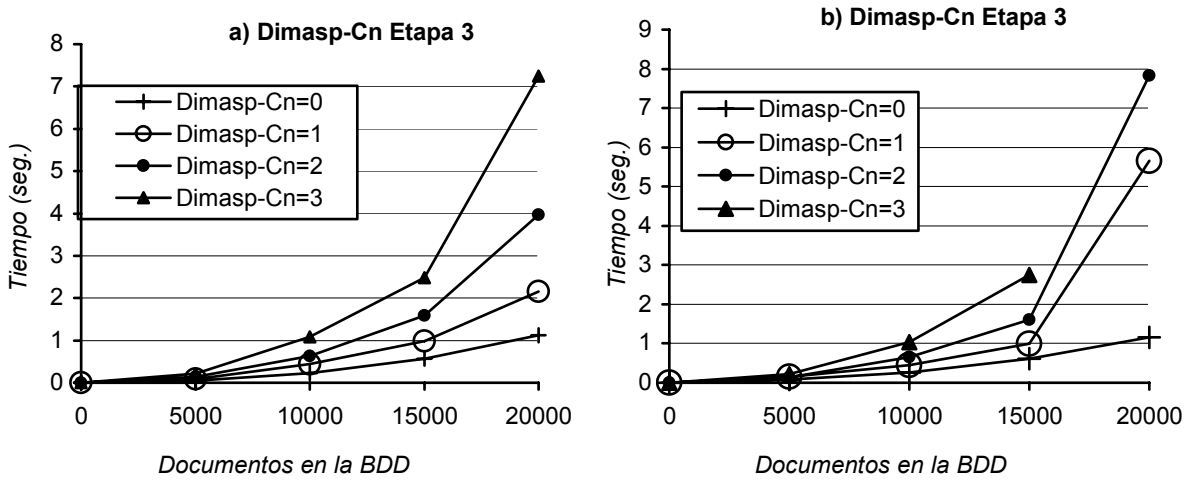


Figura 3.52 Tiempos obtenidos para la tercera etapa de Dimasp-C_n con y sin escritura de archivos temporales con $\beta = 15$ vairando el GAP entre 0 y 3.

La figura 3.53(a) muestra los tiempos correspondientes a la cuarta etapa sin escritura de archivos, en la cual, para efectos de visualización, se omitieron los 167 y 686 segundos requeridos en 20,000 documentos con GAP 2 y 3, respectivamente. De igual forma se omitieron de la figura 3.53(b) los 180 y 538 segundos requeridos en 20,000 documentos con GAP 2 y 3, respectivamente. El desempeño de esta etapa se ve afectado por la gran cantidad de SF's generadas en la etapa 2, siendo una de las etapas que consume más tiempo, especialmente para valores mayores de GAP.

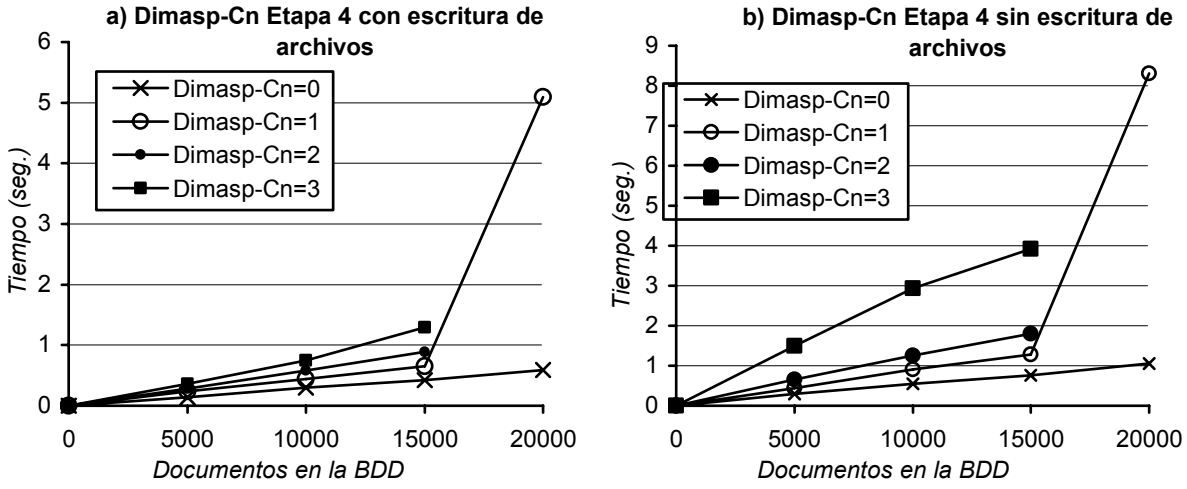


Figura 3.53 Tiempos obtenidos para la cuarta etapa de Dimasp-C_n con y sin escritura de archivos temporales con β = 15 variando GAP entre 0 y 3.

En la gráfica de la figura 3.54 se muestra el número de SFM's encontradas para GAP 0 y 1, con β = 0.1% de la cantidad de documentos en la BDD.

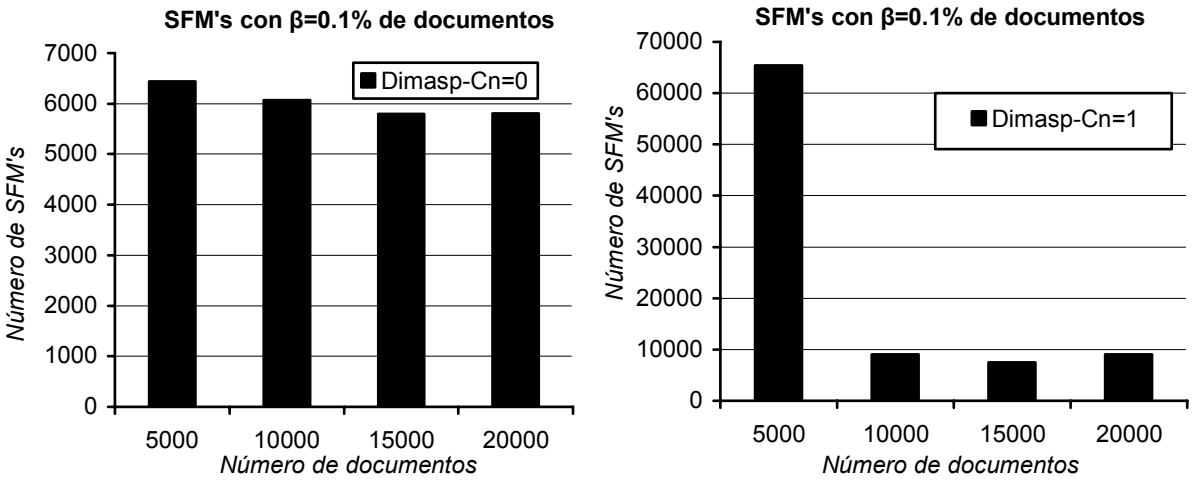


Figura 3.54 Número de SFM's con β = 0.1% de los documentos en la colección para GAP 0 y 1.

Por último, se muestra en la gráfica de la figura 3.55 el número de SFM's encontradas para GAP 2 y 3 con β = 0.1%.

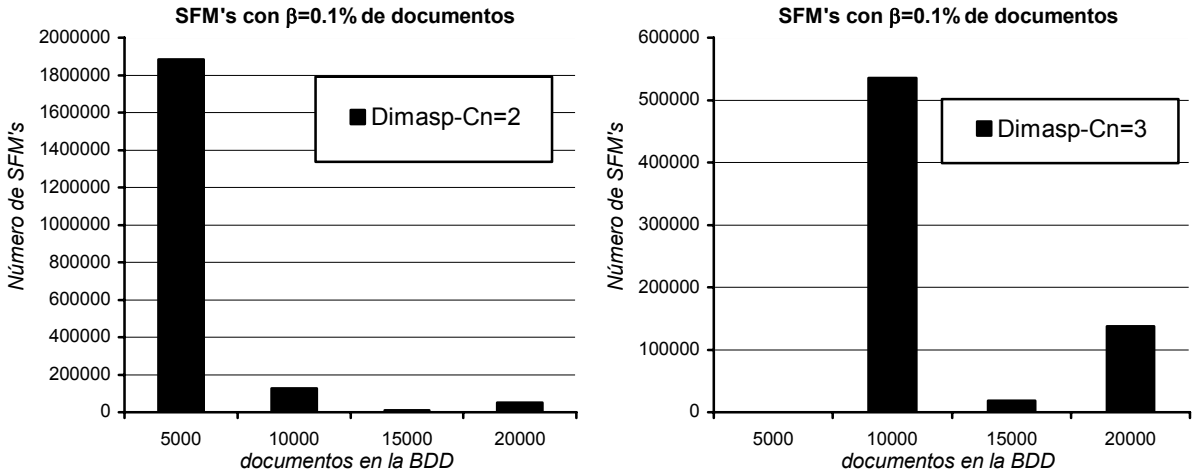


Figura 3.55 Número de SFM's con $\beta = 0.1\%$ de la cantidad de documentos en la colección, para GAP 2 y 3.

Los experimentos de las figuras 3.54 y 3.55 utilizan un umbral de frecuencia relativamente bajo pues para umbrales mayores se encuentran pocas SFM's. Esto se puede ver en la figura 3.56 que muestra el número de SFM's utilizando como umbral el 5% de la cantidad de documentos en la BDD. Cabe señalar que no fue posible procesar con Dimasp-Cn=3 para 5000 documentos con β=0.1%, debido a problemas en la memoria principal.

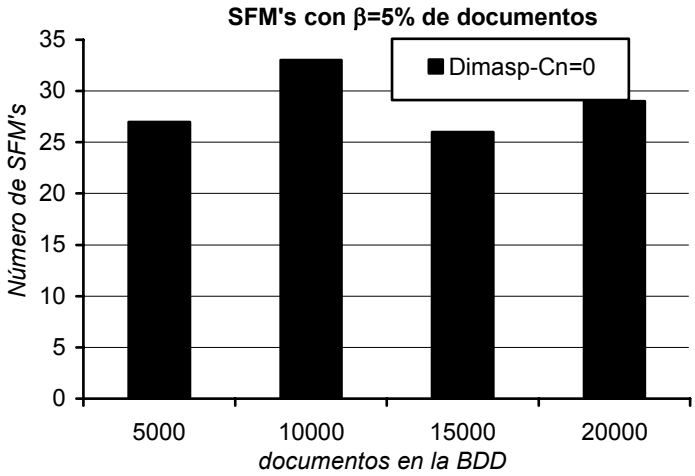


Figura 3.56 Número de SFM's con $\beta = 5\%$ de la cantidad de documentos en la colección para GAP = 0.

3.2.4 Sumario

En esta sección se definió de manera formal el problema de búsqueda de SFM's en una colección de documentos contemplando los parámetros de frecuencia mínima y GAP. Para este problema se propone el algoritmo Dimasp-C_n el cual emplea la estrategia de crecimiento de patrones evitando la generación de secuencias candidatas. Dimasp-C_n está clasificado como ascendente, de crecimiento de patrones y con búsqueda primero en amplitud. Para hacer la búsqueda, Dimasp-C_n construye una estructura de datos de orden lineal en tiempo y espacio (para GAP pequeños). Dimasp-C_n permite trabajar con diferentes umbrales de soporte sin tener que reconstruir la estructura de datos. Según las evaluaciones empíricas Dimasp-C_n supera el desempeño de otros algoritmos de minería de patrones para el descubrimiento de SFM's con GAP=n en una colección de documentos.

CAPÍTULO 4

Descubrimiento de SFM's en un sólo documento

En el capítulo anterior se presentaron los algoritmos Dimasp-C_0 y Dimasp-C_n los cuales descubren las secuencias frecuentes maximales (SFM's) a partir de una colección de documentos. Sin embargo, en varias tareas en el análisis automático de texto como clasificación de documentos, agrupamiento de documentos, resúmenes de texto, etc., se tiene la necesidad de analizar un sólo documento, por lo que Dimasp-C_0 y Dimasp-C_n no encontrarían SFM alguna. Una alternativa es dividir el documento original en oraciones, párrafos o secciones para considerarlas como la colección de documentos para Dimasp-C_n. Sin embargo, esto afecta y limita las SFM's que se pueden encontrar, en el anexo 4 se

muestran algunas de las SFM's que se pueden encontrar cuando un documento no es dividido en oraciones.

En este capítulo se presentan los algoritmos desarrollados en esta tesis para el descubrimiento de secuencias frecuentes maximales (SFM's) a partir de un solo documento con restricción GAP=0 (Dimasp-D_0) y con restricción GAP=n (Dimasp-D_n). En ambos casos, primero se plantea el problema, luego se describe el nuevo algoritmo propuesto y se muestra experimentalmente su comportamiento.

4.1. Descubrimiento de SFM's en un documento para GAP=0

En esta sección se presenta Dimasp-D_0 (***Di**scovering **Ma**ximal **S**equential **P**atterns in a single **D**ocument for GAP=**0***) que es un algoritmo para el descubrimiento de todas las secuencias frecuentes maximales con GAP=0 en un sólo documento. En la siguiente sección se plantea formalmente el problema; posteriormente, se introduce Dimasp-D_0 y finalmente se evalúa su desempeño utilizando documentos de gran longitud.

A diferencia de las SFM's encontradas en una colección de documentos, en el problema de Dimasp-D_0, se ha definido que las apariciones de una secuencia frecuente no deben traslaparse en el documento, es decir, deben ser mutuamente excluyentes. De lo contrario se obtendría al mismo documento como la SFM, sin importar el umbral β . Por ejemplo, si se tuviera el documento $\langle A,B,C,D,E,F\rangle$ con GAP=0 y no fuera necesario que las apariciones de una SF sean mutuamente excluyentes, entonces el mismo documento $\langle A,B,C,D,E,F\rangle$ sería la SFM resultante, sin importar el umbral β. Esto sucede porque la SFM completa se repite β veces en la misma posición, es decir, las β apariciones de la SFM empiezan en la posición 1 (letra A) y terminan en la posición 6 (letra F). Por tal motivo, se buscan SF's tales que no compartan palabras, por lo que la posición final de una aparición de una SF debe ser estrictamente menor a la posición inicial de la siguiente aparición de dicha secuencia.

El caso anterior muestra la necesidad de que las apariciones de las SF's sean mutuamente excluyentes. Esta restricción implica la necesidad de establecer los límites que permiten obtener las SFM's más largas de acuerdo al umbral β, además de ser mutuamente excluyentes. Por ejemplo, supongamos que el documento es $\langle A,A,A,A,A,A,A,A,A,A\rangle$, β=2 y GAP=0. En este caso, la SFM es $\langle A,A,A,A,A\rangle$. Para verificar que la secuencia $\langle A,A,A,A,A\rangle$ está contenida al menos β veces en el documento se debe tomar la primera secuencia de la 1ª

hasta la 5ª posición, y la segunda secuencia de la 6ª a la 10ª posición; lo cual permite tener la SF $\langle A,A,A,A,A \rangle$ como una SFM. La figura 4.1 ilustra los límites de estas secuencias en el documento.

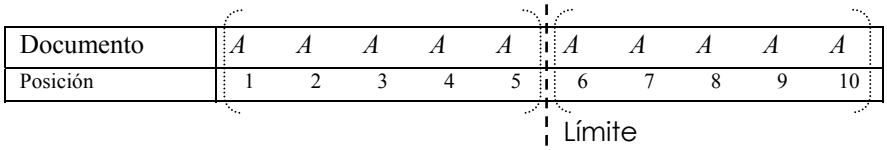


Figura 4.1 SFM's para el documento $\langle A,A,A,A,A,A,A,A,A,A \rangle$ con umbral $\beta = 2$.

Sin embargo, si cambiamos el umbral a β=3 en el ejemplo anterior, la SFM sería $\langle A,A,A \rangle$ y los límites que permiten obtener las SF's más largas para ese β son los presentados en la figura 4.2. Puede notarse que con β=3 existen cuatro opciones para considerar los límites de las SF's.

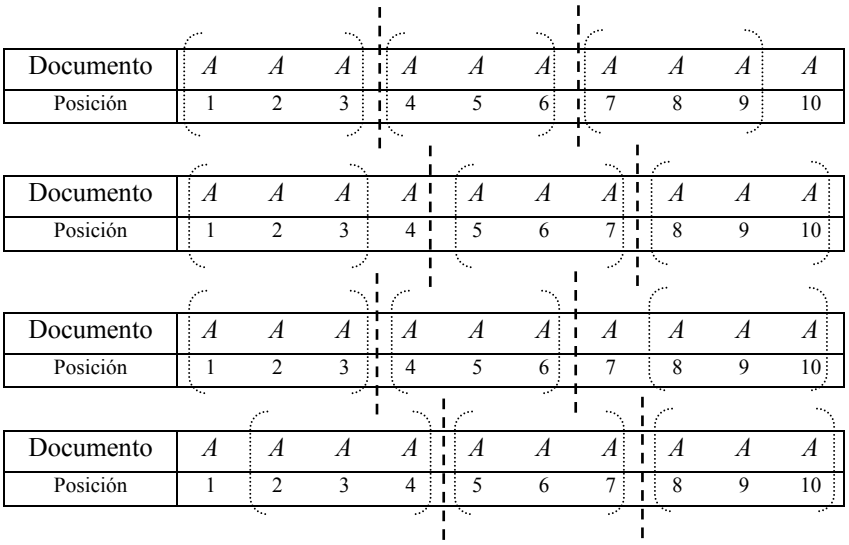


Figura 4.2 SFM's para el documento $\langle A,A,A,A,A,A,A,A,A,A \rangle$ con umbral $\beta = 3$.

Los ejemplos anteriores muestran que para encontrar los límites de acuerdo al umbral β existen diferentes opciones. A continuación se ilustra la existencia de casos donde las SFM's no son tan claras a simple vista. Por ejemplo, para el documento $\langle E,S,E,S,E \rangle$ con β=2 y GAP=0, las SFM's serían $\langle E,S \rangle$ y $\langle S,E \rangle$, como se muestra en a figura 4.3.

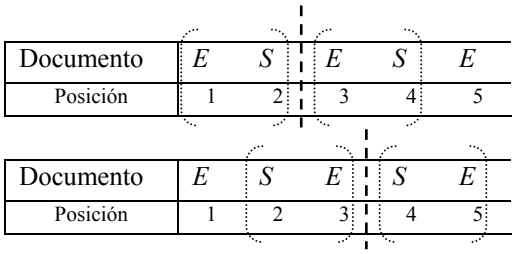


Figura 4.3 SFM's para el documento $\langle E,S,E,S,E\rangle$ con umbral $\beta = 2$.

En el ejemplo anterior se observa como al cambiar el límite es posible encontrar SFM's diferentes. A continuación se define el problema de búsqueda de SFM's en un documento. Posteriormente se describe el algoritmo propuesto para encontrar dichas secuencias.

4.1.1. Definición del problema para GAP=0

Una ***secuencia*** S, denotada por $\langle s_1,s_2,...,s_k\rangle$, es una lista ordenada de k elementos, denominada k~secuencia.

Sea $P=\langle p_1,p_2,\ldots,p_t\rangle$ y $S=\langle s_1,s_2,\ldots,s_m\rangle$ secuencias, P es una ***subsecuencia con GAP=0*** de S, denotado como $P\subseteq_0 S$, si existe un entero $i\geq 1$ tal que $p_1=s_i$, $p_2=s_{i+1}$, $p_3=s_{i+2}$, $\ldots$,$p_t=s_{i+(t-1)}$.

Un ***documento*** W se puede considerar como una secuencia de palabras, denotado también como $\langle w_1,w_2,\ldots,w_t\rangle$.

Sean $X\subseteq_0 W$ y $R\subseteq_0 W$ entonces X y R son ***mutuamente excluyentes*** si X y R no comparten elementos *i.e.*, si ($x_n=w_i$ y $r_1=w_j$) o ($r_n=w_i$ y $x_1=w_j$) entonces $i<j$.

La ***frecuencia*** de una secuencia S en un documento W considerado como secuencia, denotada por S_f o $\langle s_1,s_2,...,s_t\rangle_f$, es el número de veces que S está contenida en W de forma mutuamente excluyente.

Dado un umbral definido por el usuario (β), una secuencia S es ***frecuente*** si $S_f \geq \beta$. Una secuencia frecuente S es ***maximal*** si S no es subsecuencia de alguna otra secuencia frecuente. A una secuencia frecuente maximal (SFM) también se le conoce como ***patrón secuencial maximal***.

Dado un documento, el problema abordado en esta sección es el descubrimiento de todas las secuencias frecuentes maximales con GAP=0 para un umbral β dado.

4.1.2. Algoritmo propuesto para GAP=0 (Dimasp-D_0).

La idea básica del algoritmo de Dimasp-D_0 es similar a Dimasp-C_0, donde la búsqueda de SFM's se hace en una estructura de datos construida a partir de una colección de documentos, por lo cual en Dimasp-D_0 se buscó construir una estructura similar pero para un sólo documento, sin embargo fue necesario modificarla para garantizar que las apariciones de una SFM sean mutuamente excluyentes.

Al igual que en Dimasp-C_0, para la construcción de la estructura de datos de Dimasp-D_0 se representa cada documento mediante los pares de palabras que contiene. Por ejemplo, el documento D=⟨*My, dog, is, nice, My, cat, is, nice, My, dog, is, nice*⟩ se puede representar por sus pares de palabras ⟨*My, dog*⟩, ⟨*dog, is*⟩, ⟨*is, nice*⟩, ⟨*nice, My*⟩, ⟨*My, cat*⟩, ⟨*cat, is*⟩, ⟨*is, nice*⟩, ⟨*nice, My*⟩, ⟨*My, dog*⟩, ⟨*dog, is*⟩ y ⟨*is, nice*⟩.

Si el umbral de frecuencia β es 2 entonces, en el documento D, ⟨*My, dog, is, nice*⟩ es una *SFM* la cual también puede ser representada mediante los pares de palabras ⟨*My, dog*⟩$_2$ ⟨*dog, is*⟩$_2$ y ⟨*is, nice*⟩$_3$, los cuales tienen cada uno una frecuencia mayor o igual a β (el subíndice en el par representa la frecuencia de ese par en el documento D). Esto nos permite localizar la *SFM* encontrando primero todas las ocurrencias de la secuencia ⟨*My, dog*⟩ en el documento, donde el número de ocurrencias tiene que ser mayor o igual a β. Por ejemplo, en la figura 4.4 están señaladas las ocurrencias del par ⟨*My, dog*⟩ en el documento.

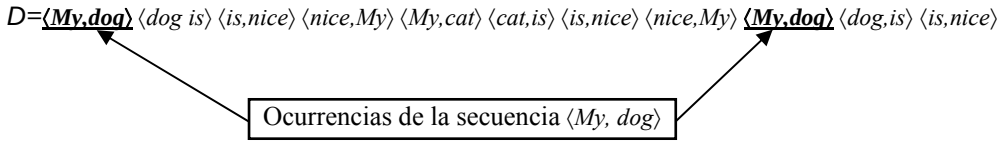


Figura 4.4 Ocurrencias de la secuencia ⟨*My, dog*⟩ en el documento.

Una vez encontrado el conjunto de todas las ocurrencias de la secuencia ⟨*My, dog*⟩, para encontrar la SFM ⟨*My, dog, is, nice*⟩ habrá que descartar aquellas ocurrencias que no continúen con el par de palabras ⟨*dog, is*⟩ (que en este ejemplo no existen). De esta manera el nuevo conjunto de ocurrencias corresponde a la secuencia ⟨*My, dog, is*⟩, lo cual se muestra en la figura 4.5.

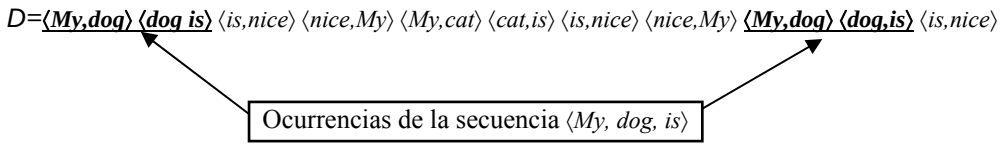


Figura 4.5 Ocurrencias de la secuencia ⟨*My, dog, is*⟩ en el documento *D*.

De manera similar, una vez encontrado el conjunto de ocurrencias de la secuencia ⟨*My, dog, is*⟩, habrá que descartar aquellas ocurrencias que no continúen con el par de palabras ⟨*is, nice*⟩, de esta manera el nuevo conjunto de ocurrencias permite encontrar la SFM ⟨*My, dog, is, nice*⟩ en el documento *D*, lo cual se muestra en la figura 4.6. Ahora, la SFM ⟨*My, dog, is, nice*⟩ no puede crecer más, pues sólo una de las secuencias continúa con el par ⟨*nice, My*⟩, por lo tanto la secuencia ⟨*My, dog, is, nice*⟩ es una SFM.

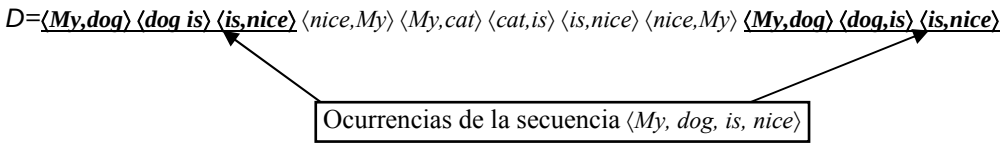


Figura 4.6 Ocurrencias de la secuencia ⟨*My, dog, is, nice*⟩ en el documento *D*.

Siguiendo esta idea, Dimasp-D_0 construye una estructura de datos a partir del documento. Para ello, se utiliza una lista (llamada *Lista* Δ) la cual liga todas las ocurrencias del par ⟨*My, dog*⟩ en todo el documento *D*, de manera que a partir de *Lista* Δ sea posible recuperar de manera rápida las posiciones donde ocurre la secuencia ⟨*My, dog*⟩. Así, la frecuencia de la secuencia ⟨*My, dog*⟩ estará determinada por el número ocurrencias del par en *Lista* Δ. La figura 4.7 ilustra la forma en que *Lista* Δ liga las ocurrencias de la secuencia ⟨*My, dog*⟩ en el documento *D*, en este caso *Lista* Δ tiene dos ocurrencias del par ⟨*My, dog*⟩.

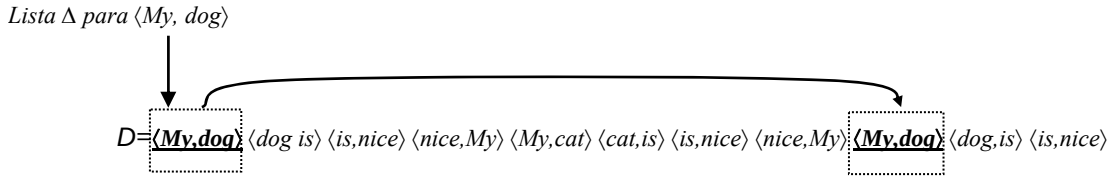


Figura 4.7 Representación de *Lista* Δ para el par de palabras ⟨*My, dog*⟩ en el documento *D*.

A partir de *Lista* Δ es posible crecer la secuencia ⟨*My, dog*⟩ a la secuencia ⟨*My, dog, is*⟩, siempre y cuando los pares de palabras ⟨*My, dog*⟩ y ⟨*dog, is*⟩ se presenten en este orden al menos β veces

en el documento. Para el ejemplo anterior, *Lista* Δ cambiaría quedando como se muestra en la figura 4.8.

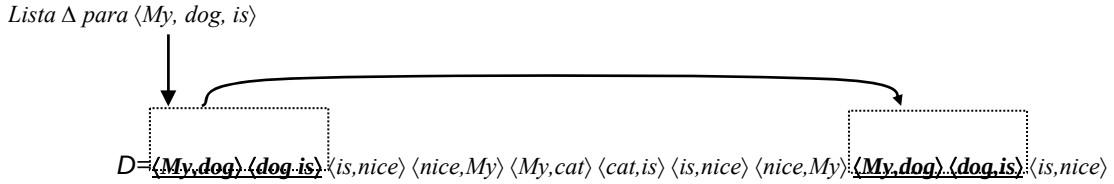


Figura 4.8 *Lista* Δ para los pares de palabras ⟨*My, dog*⟩ y ⟨*dog, is*⟩ en el documento *D*.

Utilizando la *Lista* Δ actual es posible realizar el crecimiento de la secuencia ⟨*My, dog, is*⟩ a ⟨*My, dog, is, nice*⟩, quedando la *Lista* Δ como se muestra en la figura 4.9. Finalmente la secuencia ⟨*My, dog, is, nice*⟩ ya no puede crecer más, pues es una SFM.

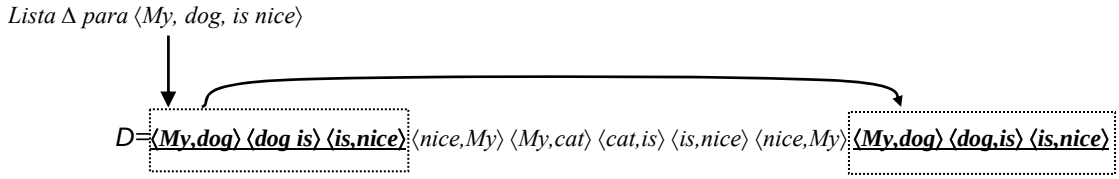


Figura 4.9 *Lista* Δ para los pares de palabras ⟨*My, dog*⟩, ⟨*dog, is*⟩ y ⟨*is, nice*⟩ en el documento *D*.

En general, Dimasp-D_0 mantiene una *Lista* Δ por cada par de palabras diferentes del documento, las cuales son almacenadas en un arreglo. En este arreglo es posible acceder a las características de cada par, esto es, el par de palabras, su frecuencia y la lista de ocurrencias en todo el documento. Utilizando el ejemplo anterior, se muestra en la figura 4.10 una representación del arreglo, para cada par de palabras del documento *D*=⟨*My, dog, is, nice, My, cat, is, nice, My, dog, is, nice*⟩.

Índice	$\langle w_i, w_{i+1} \rangle$	C_f	¿Ya fue usado el par?	Lista Δ
1	⟨My, dog⟩	2	No	⟨My, dog⟩ → ⟨My, dog⟩ →
2	⟨dog, is⟩	2	No	⟨dog, is⟩ → ⟨dog, is⟩ →
3	⟨is, nice⟩	3	No	⟨is, nice⟩ → ⟨is, nice⟩ → ⟨is, nice⟩ →
4	⟨nice, My⟩	2	No	⟨nice, My⟩ → ⟨nice, My⟩ →
5	⟨My, cat⟩	1	No	⟨My, cat⟩ →
6	⟨cat,is⟩	1	No	⟨cat,is⟩ →

Figura 4.10 Arreglo para los pares de palabras diferentes del documento *D* = ⟨*My, dog, is, nice, My, cat, is, nice, My, dog, is, nice*⟩.

Hasta este momento, se observa que la estructura de datos de Dimasp-D_0 está basada en representar el documento mediante los pares de palabras que contiene. Sin embargo, los pares no preservan su estructura secuencial en el documento, por lo que con esta estructura de datos todavía no es posible realizar el crecimiento de la secuencia frecuente. Esto puede resolverse ligando los pares de palabras en orden secuencial. En la figura 4.11 se muestra la manera en que son ligados los pares de palabras. También se observa en la figura 4.11 que cada par tiene un número asociado el cual permite ordenar los pares consecutivamente, y se utilizará para garantizar que las secuencias frecuentes se presenten de manera mutuamente excluyente.

índice	$\langle w_i, w_{i+1} \rangle$	C_f	¿Ya fue usado el par?	Lista Δ
1	⟨My, dog⟩	2	No	
2	⟨dog, is⟩	2	No	
3	⟨is, nice⟩	3	No	
4	⟨nice, My⟩	2	No	
5	⟨My, cat⟩	1	No	
6	⟨cat,is⟩	1	No	

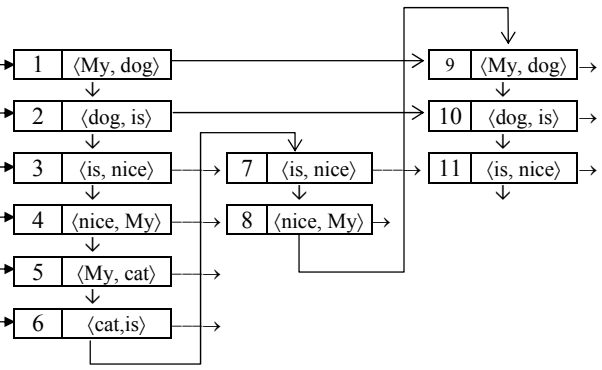


Figura 4.11 Estructura de datos construida en Dimasp-D_0 en la cual están ligados los pares de palabras contiguos, para el documento $D = \langle My, dog, is, nice, My, cat, is, nice, My, dog, is, nice \rangle$.

Con el propósito de reducir la información almacenada en *Lista* Δ, es posible sustituir los pares de palabras por el índice donde se encuentra el par en el arreglo, lo cual se muestra en la figura 4.12.

índice	$\langle w_i, w_{i+1} \rangle$	C_f	¿Ya fue usado el par?	Lista Δ
1	⟨My, dog⟩	2	No	
2	⟨dog, is⟩	2	No	
3	⟨is, nice⟩	3	No	
4	⟨nice, My⟩	2	No	
5	⟨My, cat⟩	1	No	
6	⟨cat,is⟩	1	No	

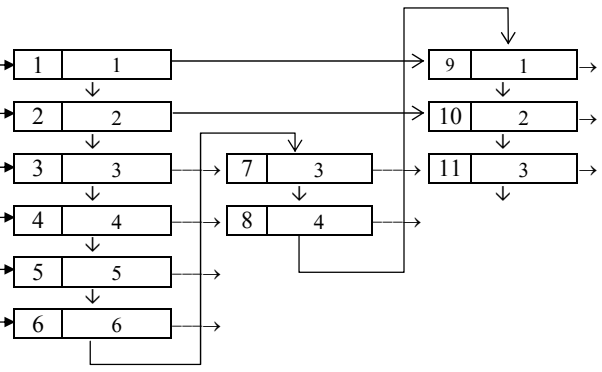
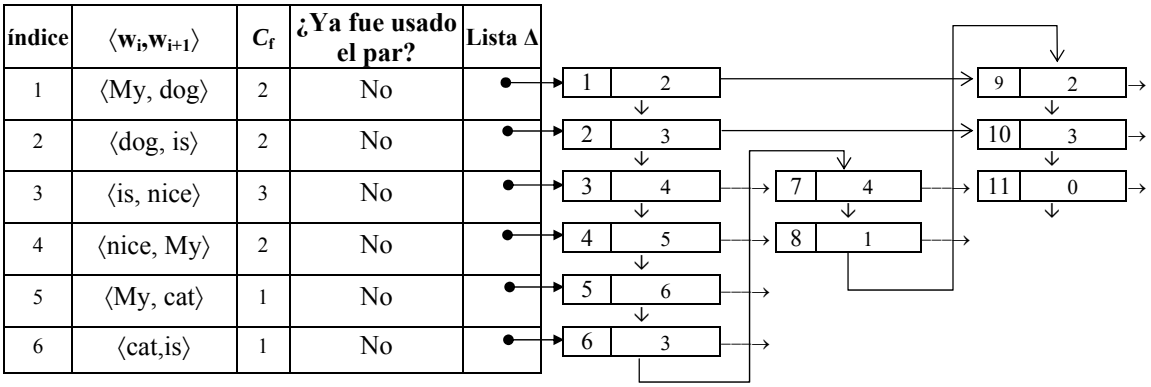


Figura 4.12 Estructura de datos construida en Dimasp-D_0 en la cual se sustituyó el par de palabras por el índice en el arreglo donde aparece ese par de palabras.

La estructura de la figura 4.12 es básicamente la estructura de datos empleada en Dimasp-D_0, sin embargo, se puede cambiar el índice donde aparece un par de palabras en el arreglo por el índice donde aparece el siguiente par en el arreglo, de esta manera se puede conocer más rápido cuál es el siguiente par de palabras. La figura 4.13 muestra como quedaría finalmente la estructura de datos empleada por Dimasp-D_0.



índice	$\langle w_i, w_{i+1} \rangle$	C_f	¿Ya fue usado el par?	Lista Δ
1	⟨My, dog⟩	2	No	
2	⟨dog, is⟩	2	No	
3	⟨is, nice⟩	3	No	
4	⟨nice, My⟩	2	No	
5	⟨My, cat⟩	1	No	
6	⟨cat,is⟩	1	No	

Figura 4.13 Estructura de datos construida en Dimasp-D_0 en la cual se cambió el índice donde aparece un par de palabras en el arreglo por el índice donde aparece el siguiente par en el arreglo.

Como se puede observar, la idea básica del algoritmo consiste en encontrar todas las SF's en una estructura de datos, la cual guarda todos los pares distintos de palabras consecutivas que aparecen en el documento, sin perder el orden secuencial. Dado un umbral β especificado por el usuario, el algoritmo revisa en el arreglo si algún par de palabras tiene una frecuencia mayor o igual a β. En tal caso, ese par de palabras es una secuencia frecuente, por lo tanto habrá que determinar, para cada par de *Lista* Δ, la SF más larga que se puede formar según β. En este crecimiento es necesario asegurar que todas las SF's sean mutuamente excluyentes. Una vez que no es posible el crecimiento de una SF, debido a que deja de ser frecuente, entonces se obtiene una SF que probablemente es maximal. En este sentido, una SF será una SFM si no es subsecuencia de alguna otra secuencia frecuente construida a partir de otro par. Es por eso que después de haber generado todas las SF's más largas, para todos los pares de palabras, se seleccionan aquellas que son maximales.

De manera similar a Dimasp-C_0, el algoritmo Dimasp-D_0 está dividido en 4 etapas. En la primera etapa se transforman y preparan los datos para la segunda etapa. En la segunda etapa se construye la estructura de datos a partir de la cual se realizará la búsqueda de SF's. La primera y segunda etapa de Dimasp-D_0 se realiza de manera muy similar a Dimasp-C_0, sin embargo es necesario considerar información adicional, la cual se describe en la siguiente

sección. En la tercera etapa, se desarrolla la búsqueda para encontrar todas las SF's más largas para cada par de palabras frecuente, de acuerdo al umbral especificado por el usuario (β). Por último, en la cuarta etapa a partir del conjunto de SF's encontradas se seleccionan las SFM's. Cabe señalar, que esta última etapa se realiza de la misma manera que la cuarta etapa de Dimasp-C_0 descrita en la sección 3.1.2.4.

4.1.2.1 Primera etapa, transformación del documento

La idea principal de la primera etapa del algoritmo consiste en asignar un identificador numérico a cada palabra diferente en el documento. Junto con el identificador asignado a cada palabra se guarda la frecuencia de la palabra en el documento. Esto se hace con el objetivo de usar, en las siguientes etapas, los identificadores numéricos en lugar de las palabras del documento. También en esta etapa se asignan identificadores numéricos a los diferentes pares de palabras, con el objetivo de ubicarlos en el arreglo de la figura 4.13.

Como se aprecia, esta etapa es muy similar a la primera etapa de Dimasp-C_0 de hecho se utiliza el mismo algoritmo de la primera etapa pero además se considera la posición de cada par de palabras.

4.1.2.2 Segunda etapa, construcción de la estructura de datos

En la segunda etapa, se construye una estructura de datos a partir del documento, en la cual son almacenados todos los pares contiguos de palabras $\langle w_i, w_{i+1} \rangle$ que aparecen en el documento, además de almacenar información adicional que ayuda a preservar el orden secuencial del documento. La estructura de datos (ver figura 4.13) está compuesta de un ***arreglo*** que contiene en cada celda C el par de palabras $\langle w_i, w_{i+1} \rangle$ del documento, la ***frecuencia*** del par (C_f), una marca Booleana que indica si ese par ya fue ***usado*** (en la etapa 3) y una lista *Lista* Δ de nodos δ. Un nodo δ (ver figura 4.14) almacena la posición ($\delta.Pos$) del par $\langle w_i, w_{i+1} \rangle$ dentro del documento, el ***índice*** ($\delta.index$) de la celda donde se encuentra el par $\langle w_{i+1}, w_{i+2} \rangle$ en el arreglo, una liga ($\delta.SamePair$) para mantener la *Lista* Δ donde están los pares idénticos y una liga ($\delta.NextPair$) para preservar el orden secuencial de los pares. Por lo tanto, el número de veces que aparece un par de palabras en el documento está registrado en C_f y las posiciones en las que aparece en el documento están registradas en los nodos δ en *Lista* Δ.

La segunda etapa trabaja como sigue: para cada par de palabras $\langle w_i,w_{i+1}\rangle$ en el documento se crea un nodo $\delta(\langle w_i,w_{i+1}\rangle)$ correspondiente al par de palabras $\langle w_i,w_{i+1}\rangle$ y en $\delta(\langle w_i,w_{i+1}\rangle).Pos$ se guarda la posición de ese par en el texto y en $\delta(\langle w_i,w_{i+1}\rangle).index$ se guarda el *índice* del arreglo donde se encuentra el par $\langle w_{i+1},w_{i+2}\rangle$ (calculado previamente en la etapa 1). Cada nodo $\delta(\langle w_i,w_{i+1}\rangle)$ es agregado al final de *Lista* Δ, la cual es determinada por el *índice* del par $\langle w_i,w_{i+1}\rangle$ (también calculado previamente en la etapa 1), siempre que se agrega un nodo δ a *Lista* Δ se incrementa la frecuencia (C_f) de esa celda. Luego se ligan todos los nodos δ de manera que $\delta(\langle w_i,w_{i+1}\rangle)$ esté asociado (utilizando $\delta.NextPair$) con el nodo $\delta(\langle w_{i+1},w_{i+2}\rangle)$, a esta asociación la podemos denotar como $\delta(\langle w_i,w_{i+1}\rangle)\rightarrow\delta(\langle w_{i+1},w_{i+2}\rangle)$. En este sentido, $\delta.NextPair$ permite preservar el orden secuencial de los pares. El algoritmo de la segunda etapa se muestra a la figura 4.15, el cual permite construir la estructura de datos mostrada en la figura 4.13.

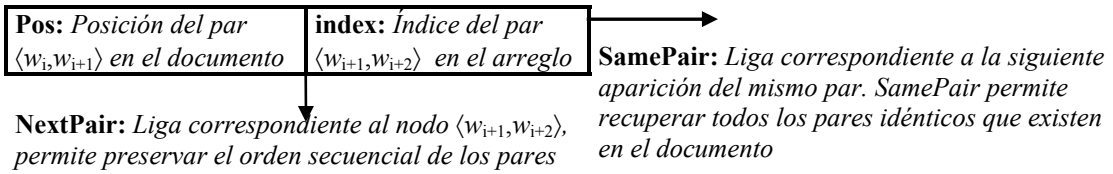


Figura 4.14 Estructura de un nodo δ.

Como se aprecia en la figura 4.15, el algoritmo para construir la estructura para Dimasp-D_0 es muy similar al de Dimasp-C_0, sólo cambia la información almacenada en los nodos δ y la manera de incrementar la frecuencia en cada celda. De igual forma, si n es el número total de palabras en el documento y m el número de pares de palabras diferentes, la segunda etapa de Dimasp-D_0 es de tiempo $\Theta(n)$ y de espacio $\Theta(n+m)$, donde $m<n$.

Etapa 2: Algoritmo para construir la estructura de datos a partir de un documento

Entrada: Un documento (D) **Salida:** El arreglo (*Array*)

Para *todos los pares de palabras* $\langle w_i,w_{i+1}\rangle \in D$ **hacer**

$\delta_i(\langle w_i,w_{i+1}\rangle) \leftarrow$ *Crear un nuevo nodo* δ

$\delta_i(\langle w_i,w_{i+1}\rangle).Pos \leftarrow i$ *//Guardar la posición inicial del par*

$\delta_i(\langle w_i,w_{i+1}\rangle).index \leftarrow$ Array[$\langle w_{i+1},w_{i+2}\rangle$] *//índice en el arreglo de la celda* $\langle w_{i+1},w_{i+2}\rangle$*, calculado en etapa 1*

index $\leftarrow$ Array[$\langle w_i,w_{i+1}\rangle$] *// índice en el arreglo de la celda* $\langle w_i,w_{i+1}\rangle$

Array[*index*].Frecuencia++ *//Incrementar en uno la frecuencia de la celda donde aparece* $\langle w_i,w_{i+1}\rangle$

Array[*index*].*Lista* $\Delta \leftarrow \delta_i(\langle w_i,w_{i+1}\rangle)$ *Agregar al final de la lista* Δ *usando* $\delta.SamePair$

$\delta_i(\langle w_i,w_{i+1}\rangle).NextPair \leftarrow \delta_{i+1}(\langle w_{i+1},w_{i+2}\rangle)$ *// Establecer la asociación* $\delta_x(\langle w_x, w_y\rangle) \rightarrow \delta_y(\langle w_y, w_z\rangle)$,

fin-para

Figura 4.15 Algoritmo para construir la estructura de datos de Dimasp-D_0.

4.1.2.3 Tercera etapa, búsqueda de SFM's de acuerdo al umbral β

En la tercera etapa, la estructura de datos construida a partir del documento es utilizada para hacer el descubrimiento de todas las SFM's con GAP=0, de acuerdo al umbral β especificado.

La tercera etapa trabaja de la siguiente manera: para cada una de las celdas $\langle w_x,w_y\rangle$ del arreglo de la estructura de datos, se verifica si la celda tiene una frecuencia (C_f) mayor o igual que el umbral β, de ser así se buscan todas las secuencias frecuentes (*SF's*) que puedan existir con el prefijo $\langle w_x,w_y\rangle$. En otras palabras, cuando se encuentra una celda $\langle w_x,w_y\rangle$ frecuente entonces a partir de los nodos $\delta(\langle w_x,w_y\rangle)$ de su *Lista* Δ se encuentran las *SF's* más largas que contengan al par $\langle w_x,w_y\rangle$ como prefijo. A partir del conjunto de nodos $\delta\langle w_x,w_y\rangle$ en *Lista* Δ, es posible determinar si la $SF=\langle w_x,w_y\rangle$ puede crecer a $\langle w_x,w_y,w_z\rangle$, verificando si existen al menos β nodos que cumplan la asociación $\delta(\langle w_x,w_y\rangle)\rightarrow\delta(\langle w_y,w_z\rangle)$, en cuyo caso sólo los nodos $\delta(\langle w_y,w_z\rangle)$ que cumplan con la asociación formarán el nuevo conjunto de nodos δ. Cabe señalar que también debe verificarse que las *SF's* $\langle w_x,w_y,w_z\rangle$ estén de forma mutuamente excluyente, más adelante se describe como garantizar esto. En general, se puede continuar el crecimiento de la *SF* si, después de añadir la palabra w_t a la *SF* $\langle w_x,w_y,w_z,w_t\rangle$, existen al menos β nodos $\delta(\langle w_y,w_z\rangle)$ que cumplan con la asociación $\delta(\langle w_y,w_z\rangle)\rightarrow\delta(\langle w_z,w_t\rangle)$. Cuando una k~*SF* ya no puede crecer es agregada al conjunto de *SF's* si $k\geq 3$, en cuyo caso todas las celdas correspondientes a sus pares de palabras son marcadas como "usadas". Este proceso se realiza para cada nodo δ de *Lista* Δ, considerando solamente los nodos δ posteriores al que se está analizando, porque los anteriores ya fueron procesados.

Al final, una vez que todas las celdas del arreglo fueron procesadas, las celdas que no fueron marcadas como "usadas" y con una $C_f \geq \beta$ son agregadas como 2~*SFM*. Todas las palabras no contenidas en los pares frecuentes del arreglo y con una frecuencia mayor o igual a β son agregadas como 1~*SFM*. Cabe señalar que para encontrar las 1~*SFM* y 2~*SFM* se utiliza el mismo algoritmo descrito en la etapa 3.2 de Dimasp-C_0 descrito en la figura 3.19. El algoritmo de la tercera etapa para encontrar todas las *SF's* se muestra en la figura 4.16.

Etapa 3.1: Algoritmo para encontrar todas las SF's con respecto al documento D
Entrada: Un documento D de la estructura de datos y un umbral β
Salida: El conjunto de SF's con respecto a D
Para *todos los nodos* $\delta_i(\langle w_i,w_{i+1}\rangle) \in D$ **hacer**
  Si Arreglo [δ_i.*index*].*frecuencia* $\geq \beta$ **entonces** *determinar todas las SF con prefijo* $\langle w_i,w_{i+1}\rangle$
    SF ← *Arreglo* [δ_i.*index*].$\langle w_i,w_{i+1}\rangle$ *//la SF inicial es el par* $\langle w_i,w_{i+1}\rangle$
    Lista Δ′ ← { $\delta_i(\langle w_i,w_{i+1}\rangle) \in$ *Arreglo*[δ_i.*index*].*Lista Δ* }
    Lista Δ′.frecuencia← Número de nodos en *Lista Δ′, es decir,* | *Lista Δ′*|
    k← *2* ← *longitud de la SF*
    Mientras *Lista Δ′.frecuencia* $\geq \beta$ **hacer** *el crecimiento de la SF*
      Si δ_{i-1}.*index* = $\delta_{i+(k-1)}$.*index* **entonces existe un ciclo**
        SF←**Ciclo**(β , *Δ′*, *Arreglo*, δ_i , *index*)
        Si *la SF no puede crecer* **entonces** salir de "mientras"
      fin-si
      De lo contrario, *continuar con el crecimiento del patrón*
        Lista Δ′ ← { $\delta_{i+(k-1)} \in$ *Lista Δ′* | $\delta_{i+(k-2)}(\langle w_{i+(k-2)},w_{i+(k-1)}\rangle) \rightarrow \delta_{i+(k-1)}(\langle w_{i+(k-1)},w_{i+(k)}\rangle)$ }
        Lista Δ′.frecuencia← Número de nodos en *Lista Δ′, es decir,* | *Lista Δ′*|
        Si *Lista Δ′.frecuencia* $\geq \beta$ **entonces** *puede crecer la SF*
          (*i*+*k*)~*SF*←$\langle w_i,w_{i+1},\ldots,w_{i+(k-2)},w_{i+(k-1)},w_{i+(k)}\rangle$
          Arreglo [$\langle w_{i+(k-1)},w_{i+(k)}\rangle$].*marca* ← "usada"
          k ←*k* + *1*
        fin-de lo contrario
    fin-mientras
    Si *para* (*i*+*k*)~*SF,* (*i*+*k*)≥ *3* **entonces** *agregar la SF al conjunto de SF's*
      SF's ← *agregar la* (*i*+*k*)~*SF*
  fin-si
fin-para

Figura 4.16 Algoritmo para encontrar las SF's usando la estructura de datos construida en la segunda etapa.

Con el propósito de ejemplificar el funcionamiento de la tercera etapa, se utiliza como entrada la estructura construida en la figura 4.13 correspondiente al documento ⟨*My, dog, is, nice, My, cat, is, nice, My, dog, is, nice*⟩ con β = 2. Se comienza analizando la primera celda del arreglo, en este caso correspondiente a ⟨*My,dog*⟩, la cual tiene una frecuencia mayor o igual que β por lo que se busca para cada nodo δ de *Lista Δ* la *SF* más larga que se puede formar. La figura 4.17 muestra parte de la estructura de la figura 4.13 involucrada en el crecimiento de la primera *SF* para la celda ⟨*My,dog*⟩. Como se aprecia en la figura 4.17 existen dos ocurrencias (nodos δ) de la secuencia ⟨*My,dog*⟩ en las posiciones 1 y 9 en el documento.

índice	$\langle w_i, w_{i+1} \rangle$	C_f	¿Ya fue usado el par?	Lista Δ
1	⟨My, dog⟩	2	No	
2	⟨dog, is⟩	2	No	
3	⟨is, nice⟩	3	No	
4	⟨nice, My⟩	2	No	
5	⟨My, cat⟩	1	No	
6	⟨cat,is⟩	1	No	

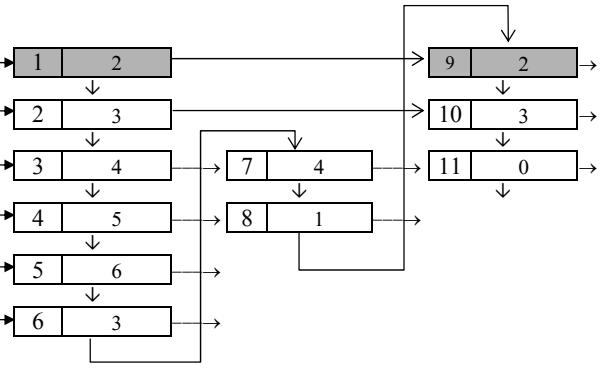


Figura 4.17 Los nodos sombreados corresponden a los nodos $\delta(\langle My,dog \rangle)$ involucrados en el crecimiento de la secuencia frecuente $\langle My,dog \rangle$.

A partir del primer nodo $\delta(\langle My,dog \rangle)$ se observa que el siguiente par de palabras es $\langle dog,is \rangle$ por lo que la $SF=\langle My,dog \rangle$ puede crecer a $\langle My,dog,is \rangle$ siempre y cuando existan al menos β nodos que cumplan con la asociación $\delta(\langle My,dog \rangle) \rightarrow \delta(\langle dog,is \rangle)$. En este caso existen 2 nodos que cumplen con $\delta(\langle My,dog \rangle) \rightarrow \delta(\langle dog,is \rangle)$, por lo que la SF crece a $\langle My,dog,is \rangle$, lo cual se muestra en la figura 4.18. Cuando la SF tiene una longitud mayor a dos, las celdas de los pares dela SF son marcadas como usadas.

índice	$\langle w_i, w_{i+1} \rangle$	C_f	¿Ya fue usado el par?	Lista Δ
1	⟨My, dog⟩	2	Si	
2	⟨dog, is⟩	2	Si	
3	⟨is, nice⟩	3	No	
4	⟨nice, My⟩	2	No	
5	⟨My, cat⟩	1	No	
6	⟨cat,is⟩	1	No	

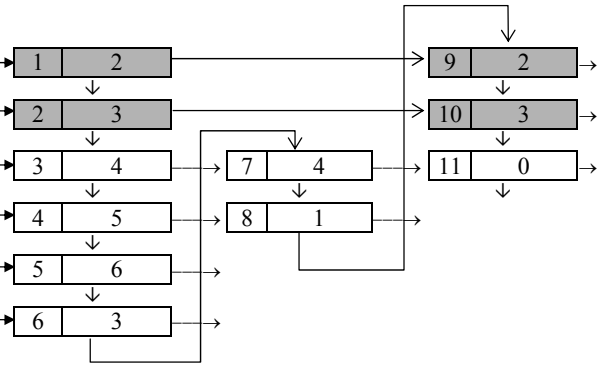


Figura 4.18 Los nodos sombreados corresponden a los nodos δ involucrados en el crecimiento de la secuencia frecuente $\langle My,dog,is \rangle$.

A partir del conjunto de nodos $\delta(\langle dog,is \rangle)$ mostrados en la figura 4.18 existen al menos β nodos que cumplen con la asociación $\delta(\langle dog,is \rangle) \rightarrow \delta(\langle is,nice \rangle)$ lo cual permite que la SF crezca a $\langle My,dog,is,nice \rangle$, esto se puede ver en la figura 4.19.

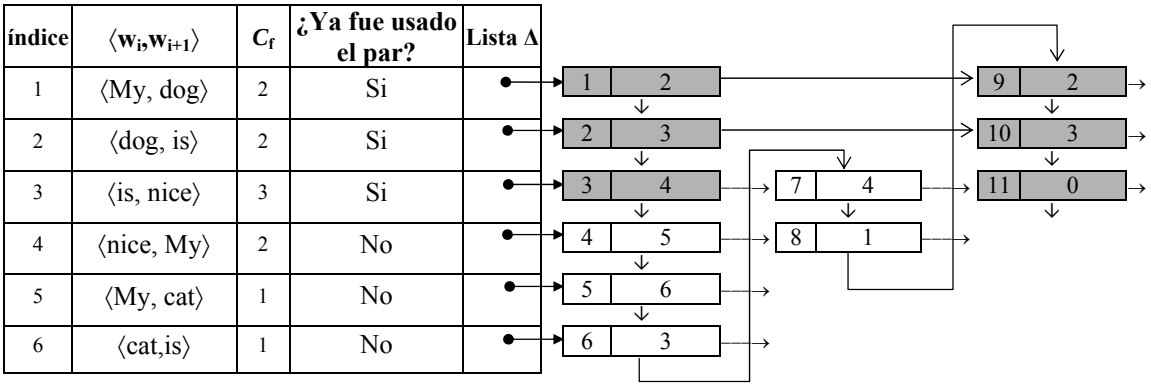


Figura 4.19 Los nodos sombreados corresponden a los nodos involucrados en el crecimiento de la secuencia frecuente ⟨*My,dog,is,nice*⟩.

En este caso, la *SF* resultante sería ⟨*My,dog,is,nice*⟩ la cual no puede crecer a ⟨*My,dog,is,nice,My*⟩ puesto que sólo un nodo δ cumple con la asociación $\delta(\langle is,nice\rangle) \rightarrow \delta(\langle nice,My\rangle)$. La *SF* resultante es agregada al conjunto de *SF's* debido a que tiene una longitud mayor o igual a 3 y por lo tanto todas las celdas del arreglo utilizadas en el crecimiento son marcadas como "usadas".

Luego se continua con el crecimiento de la siguiente *SF* del último nodo $\delta(\langle My,dog\rangle)$ en *Lista Δ*. En la figura 4.20 se muestra la parte de la figura 4.13 involucrada en el crecimiento de la *SF*=⟨*My,dog*⟩. Como se puede ver en la figura 4.20 el primer nodo $\delta(\langle My,dog\rangle)$ de la posición 1 ya no es tomado en cuenta pues ya fue procesado. Debido a que sólo un nodo $\delta(\langle My,dog\rangle)$ cumple con $\delta(\langle My,dog\rangle) \rightarrow \delta(\langle dog,is\rangle)$ la *SF* no puede crecer pues debe haber al menos $\beta = 2$ nodos. En este caso la *SF* ⟨*My,dog*⟩ tiene una longitud de 2 y por lo tanto no es agregada todavía al conjunto de SF's.

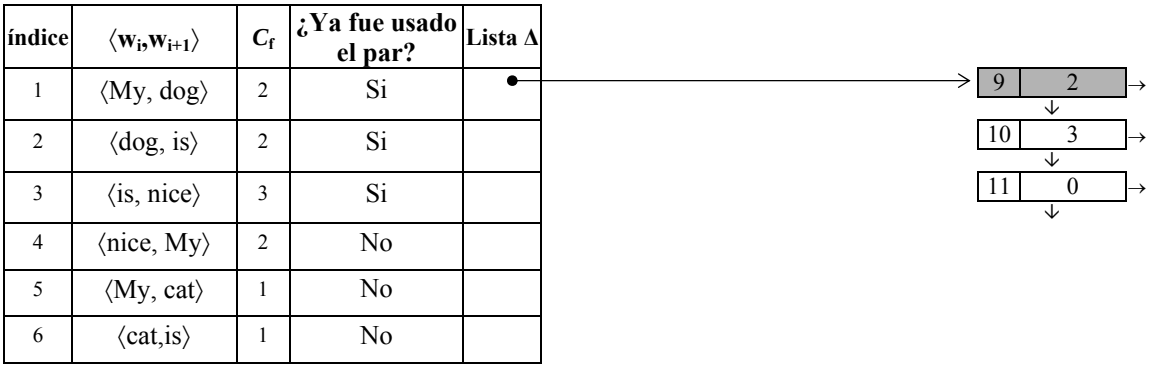


Figura 4.20 Los nodos sombreados corresponden a los nodos δ involucrados en el crecimiento de la secuencia frecuente ⟨*My,dog,is,nice*⟩, a partir del segundo nodo δ de *lista Δ* del par ⟨*My,dog*⟩.

Dimasp-D_0 continua con este proceso de manera similar para las demás celdas del arreglo, produciendo para la segunda celda la $SF\ \langle dog,is,nice\rangle$, para la tercera celda la $SF\ \langle is,nice,My\rangle$ y para la cuarta celda la $SF\ \langle nice,My\rangle$. De esta manera el conjunto de *SF's* obtenido en la tercera etapa sería $\{\langle My,dog,is,nice\rangle, \langle dos,is,nice\rangle, \langle is,nice,My\rangle\}$ y no se producirían $1\sim SFM's$ o $2\sim SFM's$ debido a que todas las celdas que tienen una frecuencia mayor o igual a β fueron marcadas como "usadas". Sería en la cuarta etapa cuando se determina el conjunto de SFM's resultantes, que es $\{\langle My,dog,is,nice\rangle, \langle is,nice,My\rangle\}$.

Aunque difícil que ocurra en texto, en el crecimiento de una *SF* se presenta un caso especial cuando una ocurrencia de la *SF* alcanza a la siguiente, es decir, el último elemento de una ocurrencia de la *SF* puede traslaparse con el primer elemento de la siguiente ocurrencia, dejando de ser mutuamente excluyentes. Por ejemplo, consideremos un umbral de frecuencia $\beta = 2$ y el documento $D=\langle A_1,B_2,C_3,A_4,B_5,C_6,A_7,B_8,C_9,A_{10},B_{11},C_{12}\rangle$, el cual tiene asociado en cada palabra la posición de dicha palabra en el documento. En este ejemplo se tendría la $SF=\langle A,B,C\rangle$ con las ocurrencias $\langle A_1,B_2,C_3\rangle$, $\langle A_4,B_5,C_6\rangle$, $\langle A_7,B_8,C_9\rangle$ y $\langle A_{10},B_{11},C_{12}\rangle$ en *D*, las cuales se muestran en la figura 4.21. Según el algoritmo de la tercera etapa la $SF=\langle A,B,C\rangle$ crecería a $SF=\langle A,B,C,A\rangle$, sin embargo las ocurrencias $\langle A_1,B_2,C_3,A_4\rangle$, $\langle A_4,B_5,C_6,A_7\rangle$ y $\langle A_7,B_8,C_9,A_{10}\rangle$ presentarían traslape, lo cual se ilustra en la figura 4.22. Cuando la ocurrencia de una SF alcanza a otra sucede que se forma una secuencia en la que la SF original se repite. Por lo cual se busca determinar cuantas veces puede repetirse la SF de forma que el resultado siga siendo frecuente sin traslaparse (mutuamente excluyentes).

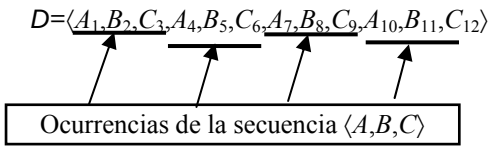


Figura 4.21 Ocurrencias de la secuencia $\langle A,B,C\rangle$ en el documento *D*.

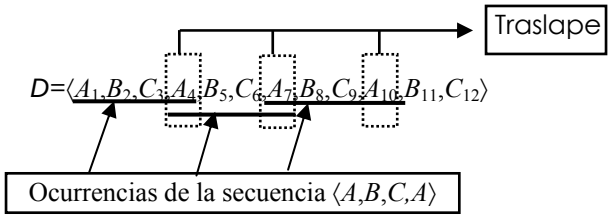


Figura 4.22 Ocurrencias de la secuencia $\langle A,B,C,A\rangle$ en el documento *D*.

Para garantizar que las ocurrencias de la *SF* sean mutuamente excluyentes se aplica el procedimiento ***ciclo***, el cual determina el mayor número de repeticiones de la SF más larga que se puede obtener, con una frecuencia mayor o igual a β. Después de este procedimiento, el crecimiento de la *SF* es llevado a cabo como se describió anteriormente. En el ejemplo anterior la SF $\langle A,B,C\rangle$ tiene las ocurrencias $\langle A_1,B_2,C_3\rangle$, $\langle A_4,B_5,C_6\rangle$, $\langle A_7,B_8,C_9\rangle$ y $\langle A_{10},B_{11},C_{12}\rangle$ de manera consecutiva, a partir de las cuales es posible encontrar la SF más larga para $\beta = 2$. Para esto se forma una secuencia a partir de todas las ocurrencias que se presentan juntas es decir, $\langle A_1,B_2,C_3\rangle$, $\langle A_4,B_5,C_6\rangle$, $\langle A_7,B_8,C_9\rangle$. y $\langle A_{10},B_{11},C_{12}\rangle$ juntas forman $\langle A,B,C,A,B,C,A,B,C,A,B,C\rangle$, esta secuencia sólo tiene una ocurrencia en D, por lo que se le puede quitar el último elemento con el objetivo de ver si aumenta el número de ocurrencias en D, lo cual tiene que hacerse hasta obtener un número de ocurrencias mayor o igual β. En este caso, quedaría la secuencia $\langle A,B,C,A,B,C\rangle$ con las ocurrencias $\langle A_1,B_2,C_3,A_4,B_5,C_6\rangle$ y $\langle A_7,B_8,C_9,A_{10},B_{11},C_{12}\rangle$, de esta manera es posible encontrar la SF más larga de acuerdo al umbral de frecuencia.

Después de aplicar el procedimiento ***ciclo***, el crecimiento de la SF continua como normalmente se haría en la tercera etapa. El algoritmo para el procedimiento ***ciclo*** se describe en la figura 4.23.

Etapa 3.1 Procedimiento ciclo: Algoritmo para encontrar la SF con ciclos
Entrada: Umbral β, documento D*;* **Salida:** *Una SF*

Intervalos ← Del conjunto de nodos $\delta(\langle w_k,w_{k+1}\rangle)$ determinar aquellas secuencias que aparecen juntas en D
$L \leftarrow j-i+1$ ← Tamaño del primer intervalo $[i, j]$
Mientras $L \geq 2$ **hacer**
  Para *cada Intervalo [Izq,Der], obtener cuantas veces cabe L* **hacer**
    TamañoIntervalo ← *Der* – *Izq* + 1
    frecuencias ← *frecuencias* + $\lceil L$ / *TamañoIntervalo* $\rceil$
    fin-para
  Si *frecuencias* ≥ β **entonces**
    $SF \leftarrow \langle w_i,\ldots,w_{i+L}\rangle$
    Si *L* no fue decrementado **entonces**
      Salir de la función, se termina el crecimiento de la SF
    De lo contrario
      $L \leftarrow L-1$
  fin-si
fin-mientras
fin

Figura 4.23 Algoritmo para encontrar la SF's cuando existen ciclos.

4.1.3 Experimentación

Con el objetivo de mostrar el tiempo de procesamiento de Dimasp-D_0 con respecto al número de palabras en un documento, se extrajeron las SFM's usando Dimasp-D_0 con un umbral $\beta = 15$. En el primer caso se escogió el documento *"Autobiography"* por Thomas Jefferson de la colección Alex [ALEX] el cual tiene 243115 caracteres, 31517 palabras, aproximadamente 100 páginas, 5499 palabras diferentes y 18739 pares diferentes. En este documento no se eliminaron *stop-words*, sólo se omitieron números y símbolos de puntuación. En el experimento de la figura 4.24 se utilizaron incrementos de 5000 palabras. De esta manera, es posible observar cómo crece el tiempo de procesamiento cuando es incrementado el número de palabras a procesar. En la figura 4.24 se muestra también el tiempo que se requiere para construir la estructura de datos y el tiempo requerido para hacer el descubrimiento de todas las SFM's. Cabe hacer notar que, una vez que la estructura de datos es construida, entonces ésta puede ser utilizada para diferentes umbrales, sin tener que reconstruirla nuevamente.

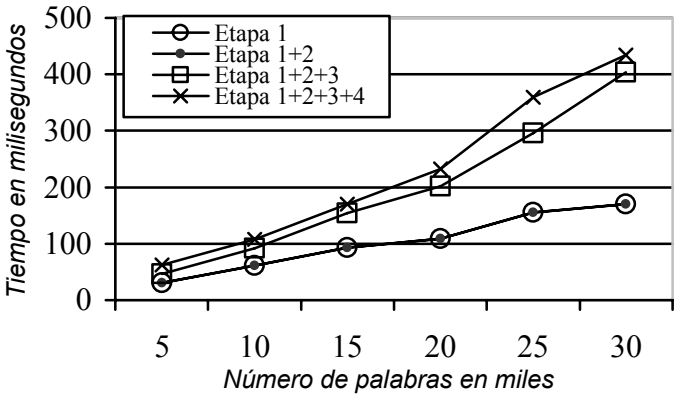


Figura 4.24 Tiempo de procesamiento para *"Autobiography"* tomando incrementos de 5000 palabras.

De manera similar al experimento anterior, en el siguiente experimento se utiliza uno de los documentos más grandes de la colección Alex, *"Letters"* por Thomas Jefferson con 1,812,428 caracteres y 305,187 palabras, aproximadamente 800 páginas, 17002 palabras diferentes y 129621 pares diferentes. En este documento, tampoco se eliminaron *stop-words*, sólo se omitieron números y símbolos de puntuación. La gráfica 4.25 muestra el tiempo de procesamiento con $\beta = 15$ cuando son procesadas las primeras 40,000 palabras, luego las primeras 80,000 palabras, y así sucesivamente hasta 280,000 palabras; es decir, se utilizaron incrementos de 40,000 palabras.

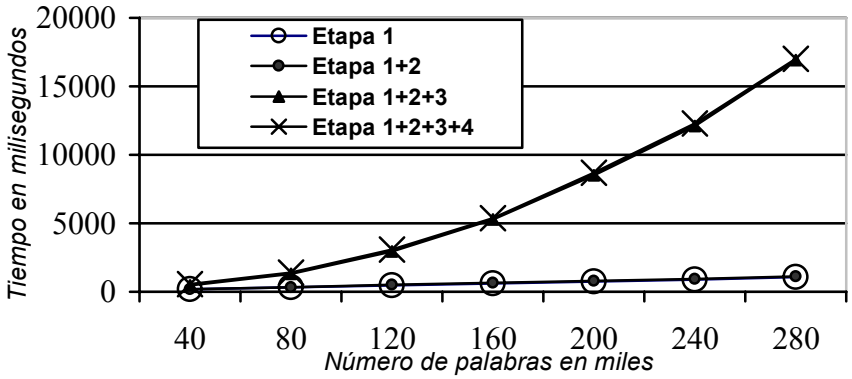


Figura 4.25 Tiempo de procesamiento para *"Letters"* tomando incrementos de 40000 palabras.

Con el fin de apreciar el comportamiento del algoritmo se hicieron experimentos variando el umbral de frecuencia β entre 2 y 25, con los documentos *"Letters"* y *"Autobiography"*. La figura 4.26a muestra el tiempo para *"Autobiography"* y la figura 4.26b para *"Letters"*, ambos con todas sus palabras.

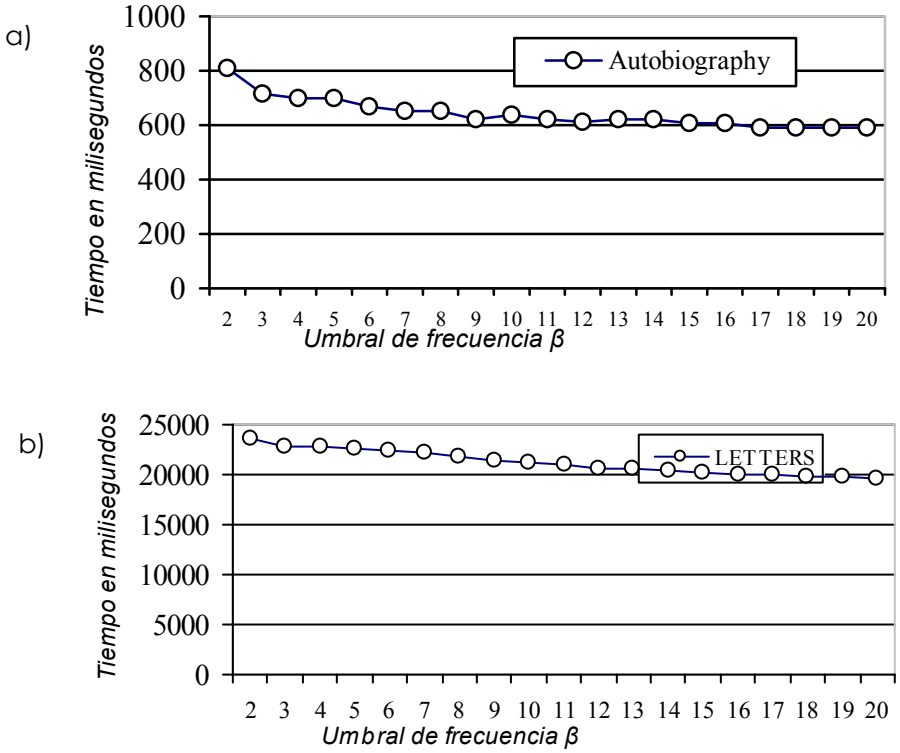


Figura 4.26 Tiempo de procesamiento de Dimasp-D_0 para diferentes valores de β: a) *"Autobiography"* y b) *"Letters"*.

En la figura 4.26 se aprecia como para umbrales pequeños (entre 2 y 5) el tiempo de procesamiento aumenta muy poco con respecto a umbrales mayores. También se incluye en la figura 4.27 una gráfica del crecimiento del número de SFM's obtenidas con diferentes umbrales y con diferentes cantidades de palabras analizadas en el documento.

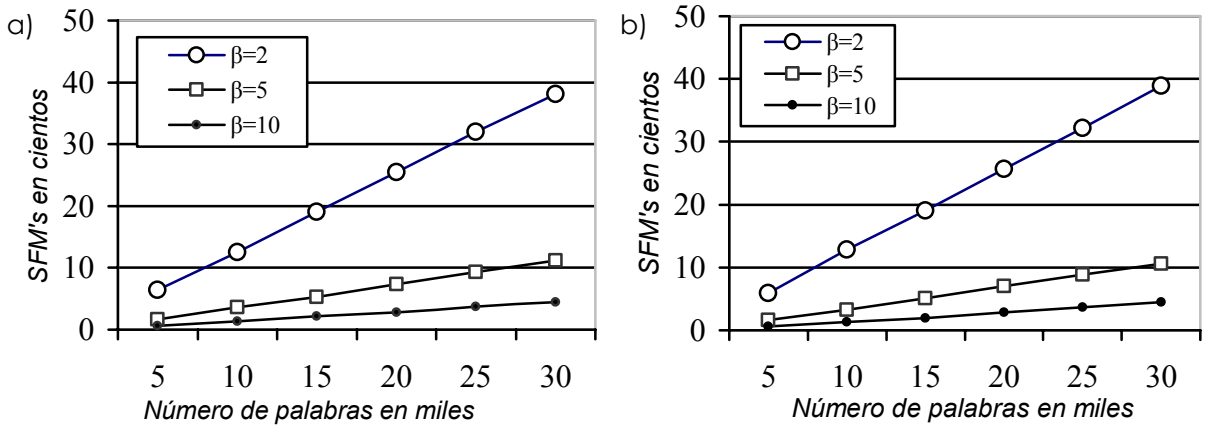


Figura 4.27 Cantidad de SFM's encontradas con diferentes valores de β para: a) *"Autobiography"* y, b) *"Letters"*.

Adicionalmente, se procesó el documento más grande de la colección Alex, *"An Inquiry into the nature ..."* por Adam Smith con 2,266,784 caracteres correspondientes a 306,156 palabras (aproximadamente 1000 páginas). En este experimento se utilizó el umbral más bajo, $\beta = 2$, Dimasp-D_0 encontró todas las 43,437 SFM's en 69 segundos.

Dimasp-D_0 fue programado en *Builder C++ versión 6*. Los experimentos aquí mostrados fueron ejecutados sobre una laptop Centrino Duo a 1.6 Ghz con 1GB de memoria principal (RAM).

4.1.4 Sumario

En esta sección se introdujo el algoritmo Dimasp-D_0 el cual encuentra las SFM's con GAP=0 en un solo documento de texto. Además, Dimasp-D_0 permite obtener diferentes conjuntos de SFM's con diferentes umbrales sin la necesidad de reconstruir la estructura de datos, puesto que esta estructura no es modificada durante el proceso de búsqueda y es independiente

del umbral requerido. Aunque Dimasp-D_0 fue diseñado para un documento, por sus características puede ser aplicado sobre cualquier otro tipo de secuencia de datos.

4.2. Descubrimiento de SFM's en un documento para GAP=n

Debido a que la definición del problema de búsqueda de SFM's con GAP=n en un sólo documento puede parecer sencillo a primera vista, se presentan algunos aspectos que, aunque difícil de ocurrir en texto, permiten ilustrar cómo es que se llegó a la definición del problema de buscar todas las SFM's en un sólo documento con GAP=n.

De manera similar al caso de la figura 4.1, presentado en Dimasp-D_0, se ha definido que las ocurrencias de las secuencias que forman una SF no deben traslaparse. Por ejemplo, si no se tuviera la definición anterior y se buscaran SFM's en el documento $\langle A,B,C,D,E,F\rangle$, entonces se produciría como SFM al documento mismo puesto que las repeticiones pueden empezar y terminar en el mismo lugar.

Cuando se buscan SFM's con GAP=n se puede generar un gran número de patrones. Por ejemplo, si se tuviera el documento $\langle A,B,C,D,E,F,G,H,X,X,X,A,B,C,D,E,F,G,H\rangle$ y se buscaran SFM's con β=2 y un GAP=1 entonces se generarían 21 SFM's como se muestra la figura 4.28.

Suponiendo que en Dimasp-D_n las apariciones de una SF deben de ser mutuamente excluyentes, es decir, que la posición final de una aparición de una SF debe ser estrictamente menor a la posición inicial de la siguiente aparición de dicha secuencia. Bajo esta suposición, con el documento $\langle A_1,B_2,A_3,C_4,B_5,C_6,D_7,E_8,A_9,B_{10},C_{11},D_{12},E_{13}\rangle$ (el subíndice sólo se utiliza para diferenciar la letra) con β=2 y GAP=1 se podrían tomar las apariciones de las secuencias $\langle A_1,B_2,C_4\rangle$ y $\langle A_9,B_{10},C_{11}\rangle$ para obtener la SFM $\langle A,B,C\rangle$. En este caso la secuencia $\langle A_3,B_5,C_6\rangle$ no fue tomada en cuenta puesto que no es mutuamente excluyente con la secuencia $\langle A_1,B_2,C_4\rangle$. Sin embargo, la decisión de considerar la aparición de la secuencia $\langle A_1,B_2,C_4\rangle$ en lugar de $\langle A_3,B_5,C_6\rangle$ no fue la mejor, ya que se pueden tomar las apariciones $\langle A_1,B_2,C_4,D_7,E_8\rangle$ y $\langle A_9,B_{10},C_{11},D_{12},E_{13}\rangle$ para formar la SF $\langle A,B,C,D,E\rangle$ que es realmente la maximal. El ejemplo anterior es un caso sencillo donde se trata de mostrar el gran número de combinaciones que se deben revisar para encontrar las SFM's, aumentando aún más la complejidad de la búsqueda. Con el fin de evitar esto, se ha definido que en Dimasp-D_n no sea necesario que las apariciones de las secuencias que forman la SF sean mutuamente

excluyentes, es decir, se permite que exista traslape en las ocurrencias que forman la SF. En el siguiente caso se ejemplifica la búsqueda con traslape.

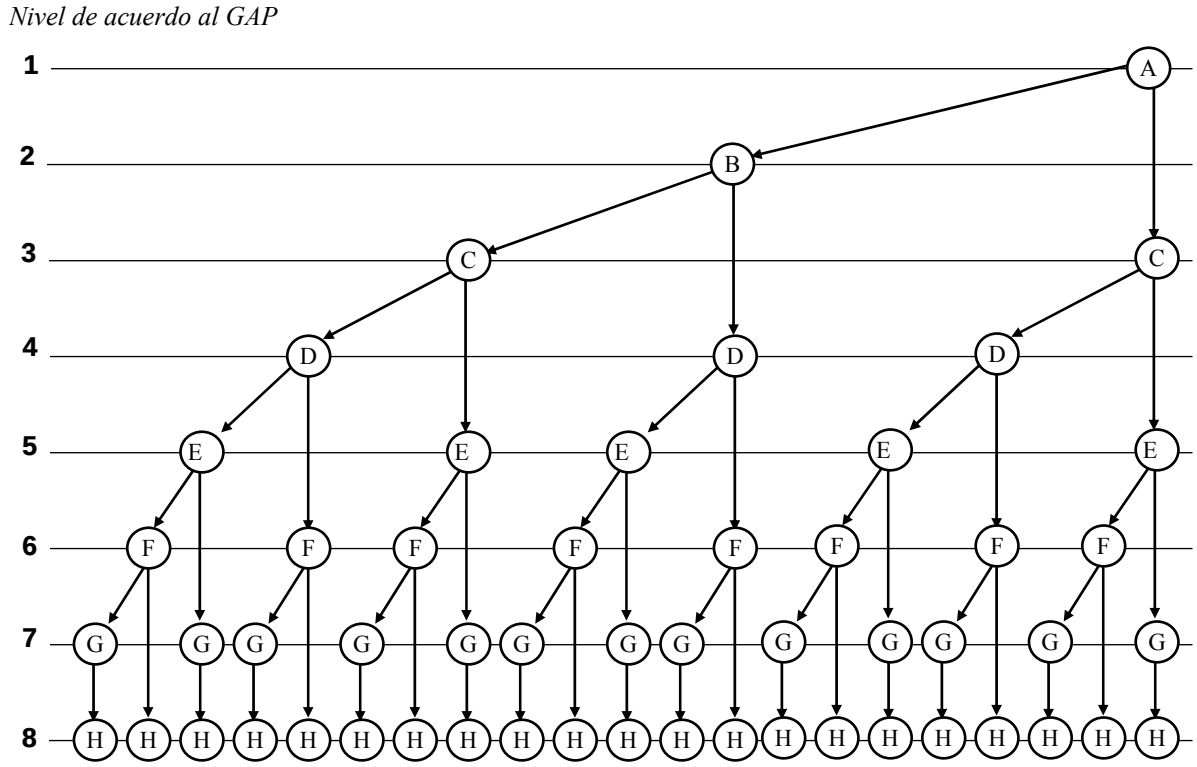


Figura 4.28 Árbol de búsqueda de SF's desarrollado para el documento $\langle A,B,C,D,E,F,G,H\rangle$.

La condición de traslape que se ha impuesto para cada ocurrencia de la SF es que deben de empezar con un par de palabras diferente, esto con el objetivo de no caer en el caso donde se puede generar el documento mismo como *SFM*, sin importar el umbral β. Por ejemplo, si se permite traslape y se tuviera el documento $\langle A_1,A_2,B_3,B_4,C_5,C_6,E_7\rangle$ con β=1 y GAP=1 se tomarían las ocurrencias $\langle A_1,B_3,C_5,E_7\rangle$, $\langle A_2,B_3,C_5,E_7\rangle$ y ($\langle A_2,B_4,C_5,E_7\rangle$ o $\langle A_2,B_4,C_6,E_7\rangle$) para formar la SF $\langle A,B,C,E\rangle$ con una frecuencia de 3. Como se aprecia en las ultimas dos ocurrencias, se pueden generar dos tipos de crecimiento, lo cual no hace más frecuente a la SF, es decir su frecuencia es de 3 y no de 4. La siguiente sección define el problema de descubrimiento de SFM's para un documento con GAP=n.

4.2.1. Definición del problema para GAP=n

Una ***secuencia*** S, denotada por $\langle s_1,s_2,...,s_k\rangle$, es una lista ordenada de k elementos denominada k~secuencia.

Sea $P=\langle p_1,p_2,...,p_t\rangle$ y $S=\langle s_1,s_2,...,s_m\rangle$ secuencias. P es una subsecuencia con ***GAP=n*** de S, denotado como $P \subseteq_n S$, si existen enteros $1\leq i_1<i_2<...<i_t\leq m$ tal que $p_1 = s_{i_1}$, $p_2 = s_{i_2}$, $p_3 = s_{i_3}$,..., $p_t = s_{i_t}$, donde $(i_k-i_{k-1})-1\leq n$. Es decir, n es la máxima separación permitida entre los elementos que forman una subsecuencia.

Un documento W se puede considerar como una secuencia de palabras, denotado como $W=\langle w_1,w_2,\ldots,w_t\rangle$.

Dado un documento W y una restricción GAP=n, la ***frecuencia*** de una secuencia S denotada por S_f, es el número de subsecuencias iguales a S en W que no comparten el primer par de palabras. En esta sección se usará el término ***secuencia*** para referirnos a las ***secuencias con*** GAP=n.

Dado un umbral de frecuencia (β), una secuencia S es ***frecuente*** si $S_f \geq \beta$. Una secuencia frecuente S es ***maximal*** si S no es subsecuencia con $\mathrm{GAP}=0$ de alguna otra secuencia frecuente. A una secuencia frecuente maximal (SFM) también se le conoce como ***patrón secuencial maximal***.

Dado un documento, el problema abordado en esta sección es el descubrimiento de todas las secuencias frecuentes maximales con GAP=n para un umbral β dado.

4.2.2. Algoritmo propuesto para GAP=n (Dimasp-D_n)

En esta sección se presenta el algoritmo Dimasp-D_n el cual permite hacer el descubrimiento de SFM's con GAP=n cuando se procesa un sólo documento. De manera similar a Dimasp-C_n, el algoritmo Dimasp-D_n está dividido en 4 etapas. En la primera etapa se transforman y preparan los datos para la segunda etapa. En la segunda etapa se construye la estructura de datos a partir de la cual se realizará la búsqueda de secuencias frecuentes. En la tercera etapa, se desarrolla la búsqueda donde son encontradas las secuencias frecuentes (*SF's*)

más largas que inician con los pares frecuentes del documento, de acuerdo al umbral especificado por el usuario (β). Por último, en la cuarta etapa se determinan cuáles de las *SF's* encontradas son en realidad las SFM's.

En general, el algoritmo Dimasp-D_n es muy similar a Dimasp-C_n, de hecho la primera etapa se realiza de la misma forma que en Dimasp-C_n (descrita en la sección 3.1.2.1).

En la segunda etapa la representación adoptada para Dimasp-D_n es básicamente la misma que para Dimasp-C_n, sin embargo ahora en lugar de guardar en cada nodo ω el identificador del documento al que pertenece dicho nodo se guarda la posición de la palabra dentro del documento.

La cuarta etapa de Dimasp-D_n se realiza de igual forma que en Dimasp-C_n (descrita en la sección 3.1.2.4).

La diferencia radica básicamente en la tercera etapa donde cambia la manera de procesar la estructura generada en la segunda etapa, la cual se describe a continuación.

4.2.2.1. Tercera etapa, búsqueda de patrones secuenciales de acuerdo al GAP y al umbral β

En la tercera etapa se utiliza la estructura de datos construida en la segunda etapa para hacer el descubrimiento de todas las SF's de acuerdo al GAP. Una vez construida la estructura de datos en la segunda etapa, en la tercera etapa de Dimasp-D_n se usa una estrategia de crecimiento por amplitud y de la identificación de subárboles iguales para encontrar todas las SFM's de acuerdo al umbral β y GAP especificados por el usuario. Para lograr esto, Dimasp-D_n hace lo siguiente: para cada celda del arreglo con frecuencia mayor o igual a β se hace el crecimiento por amplitud de cada uno de los nodos δ contenidos de la $Lista\ \Delta$, esto con el objetivo de construir un árbol similar al construido en Dimasp-C_n que determina las SF's. Para construir el árbol se utilizan nodos φ (como en Dimasp-C_n), los cuales contienen información acerca del estado de crecimiento de las SFM's, esto es, la palabra w relativa a la SF, el *nivel* del nodo de acuerdo al GAP, una $Lista\ \Omega$ que contiene ligas a nodos ω, la *frecuencia* que está determinada por el número de nodos que aparecen en $Lista\ \Omega$, además de un arreglo de *GapHijos* con GAP+1 ligas a otros nodos φ. De esta manera, la información de un nodo φ utilizada para identificar nodos iguales es w, *nivel*, $Lista\ \Omega$ y *frecuencia*.

Cabe recordar que en la segunda etapa se establece cómo son asociados los nodos de manera que el nodo ω_p está asociado (utilizando $\omega_p.GapLiga$) con el nodo $\delta(\langle w_p,w_q\rangle)$ y éste a su vez está asociado (utilizando $\delta(\langle w_p,w_q\rangle).NextPair$) con el nodo ω_q. De esta manera, un nodo $\delta(\langle w_p,w_q\rangle)$ asocia a dos nodos ω_p y ω_q lo cual podemos denotar como $\omega_p \rightarrow \delta(\langle w_p,w_q\rangle) \rightarrow \omega_q$. Sin embargo, un nodo ω_p tiene asociado GAP+1 nodos δ de manera que $\omega_p \rightarrow \delta(\langle w_p,w_1\rangle)$, $\omega_p \rightarrow \delta(\langle w_p,w_2\rangle)$,..., $\omega_p \rightarrow \delta(\langle w_p,w_{GAP+1}\rangle)$; es por eso que el crecimiento de las SF's se realiza mediante un árbol.

Para hacer el crecimiento de un nodo $\delta(\langle w_x,w_y\rangle)$ se crea el nodo $\varphi(\langle w_x,w_y\rangle)$ correspondiente a la raíz del árbol, el cual incluye como palabra $\varphi.w$ a la primera palabra (w_x) de la celda $\delta.index$, '1' como $\varphi.nivel$ del nodo y la *Lista* Ω está conformada por todos los nodos asociados ω_y que se tienen en $\delta(\langle w_x,w_y\rangle) \rightarrow \omega_y$ del resto de los nodos δ de *Lista* Δ. Sin embargo, a diferencia de Dimasp-C_n, en Dimasp-D_n cuando se agrega la palabra ω_y a la *Lista* Ω, también se agrega el identificador de la palabra $\omega_y.Id$ el cual sirve como el identificador de cada una de las secuencias que forman la SF. Es importante señalar que a partir de aquí se utilizan los identificadores de *Lista* Ω para hacer el crecimiento de las secuencias que forman la SF. De esta manera se pueden crear varias ocurrencias de la SF, pero aquellas que tengan el mismo identificador contarán sólo como una. Cabe recordar que la *frecuencia* del nodo φ está calculada con base en el número de identificadores diferentes en *Lista* Ω. Una vez incluido el nodo raíz φ como primer nivel del árbol se hace el crecimiento de las SF's; procesando todos los nodos φ de cada uno de los niveles en orden ascendente. El procesamiento de un nodo φ tiene como objetivo crear sus GAP+1 φ hijos posibles, los cuales son agregados en el árbol si son frecuentes y si no existe otro nodo idéntico a éste en el mismo nivel-gap.

A partir de un nodo $\varphi(\langle w_x,w_y\rangle)$ se puede crear un hijo $\varphi_g(\langle w_y,w_z\rangle)$ para $0 \leq g \leq GAP+1$. Para construir el nodo φ_g se asigna como la palabra $\varphi_g.w$ a w_y, como $\varphi_g.nivel$ a $\varphi.nivel+g+1$ y en $\varphi_g.Lista\ \Omega$ se agregan los nodos ω_z de la lista $\varphi.Lista\ \Omega$ que cumplan la asociación $\omega_y \rightarrow \delta(\langle w_y,w_z\rangle) \rightarrow \omega_z$. Si al finalizar de procesar el nodo φ_g se tiene una frecuencia mayor o igual que β entonces φ_g se agrega al árbol en el nivel $\varphi_g.nivel$. Siempre que se va a agregar un nodo a un nivel se verifica si existe otro idéntico a él en cuyo caso no se agrega.

Una vez procesados todos los nodos de un nivel entonces pueden ser eliminados y se continúa con el procesamiento de otro nivel. Al finalizar la construcción del árbol se generan las SF's recorriendo cada una de las ramas el árbol desde la raíz hasta las hojas. Una k~SF solamente se agrega al conjunto de SF's si $k \geq 3$, en tal caso, todas las celdas del arreglo usadas en la SF son marcadas como "usadas". Después se utiliza el algoritmo 3.2 de la

tercera etapa del algoritmo de Dimasp-C_0 (mostrado en la figura 3.19) para determinar las SFM's de tamaño 1 y 2. Después se utiliza el procedimiento de la cuarta etapa de Dimasp-C_0 (sección 3.1.2.4) para determinar cuáles de las SF's generadas son maximales. En la figura 4.29 se muestra el algoritmo de la tercera etapa para Dimasp-D_n.

Etapa 3: Algoritmo para encontrar todas las SF's
Entrada: Estructura de la etapa 2, el umbral β y el GAP **Salida:** Conjunto de SF's
$\varphi \leftarrow$Nodo del Árbol que contiene (*palabra w*, nivel, Lista Ω *de nodos* [$\omega_{1....m}$, $\omega_{1....m}$.*Id*])
Para *todos los índices de las celdas* $\in$ *Arreglo* **hacer**
  Si *Arreglo*[*índex*].$C_f \geq \beta$ **entonces** *la celda es frecuente, hacer el crecimiento de las SF's*
    Para *todos los nodos* $\delta_i \in$ *Arreglo*[*índice*].*Lista* Δ **hacer**
      $\varphi \leftarrow$ *Crear* (*Arreglo*[*índice*].w_i , 1, { [$\delta \rightarrow \omega_x$, $\delta \rightarrow \omega_x$.*Id*] $\in$ *Arreglo*[*índice*].*Lista* Δ | $\delta_i \rightarrow \omega_x$ })
      Árbol.Raíz $\leftarrow \varphi$
      Árbol.GapNivel(1) $\leftarrow$ *Árbol.Raíz*
      Para *todos los niveles-gap ng del Árbol* **hacer** *el procesamiento por amplitud*
        Para *todos los nodos* φ *del nivel ng* **hacer** *el crecimiento de nodos*
          Para $\varphi.\omega_1 \rightarrow \delta(\langle w_1, w_g \rangle) \rightarrow \omega_g$ *donde* g=1...*GAP*+1 **hacer** *el crecimiento para sus GAP*+1 *hijos*
            $\varphi_g \leftarrow$*Crear* (w_1,φ.*Nivel*+g+1,{ [ω_g ,ω_g.*Id*] $\in \varphi$.Lista Ω | $\omega_1 \rightarrow \delta(\langle w_1, w_g \rangle) \rightarrow \omega_g$}) *//crecer hijo*
            Si φ_g.*frecuencia* $\geq \beta$ **entonces** *agregar al nivel correspondiente*
              φ.*GapHijos*[g]$\leftarrow$Buscar nodo redundante (φ_g)
              Árbol.GapNivel(φ.*nivel*+g+1)$\leftarrow$ φ_g
            fin-si
          fin-para
        fin-para
      fin-para
      SF's$\leftarrow$ *Cada rama desde la raíz del Árbol a las hojas es una SF, las palabras de la SF son en* φ.*w*
      Árbol.GapNivel(*ng*+*GAP*+2) $\leftarrow$ Reutilizar memoria *Árbol.GapNivel(ng)*
    fin-para
  fin-si
fin-para

Figura 4.29 Algoritmo para encontrar las SF's que inician en cada par frecuente de la estructura de datos construida en la segunda etapa.

4.2.3 Experimentación

Dimasp-D_n fue programado en Builder C++ versión 6. Con el objetivo de permitir trabajar con documentos más grandes se decidió que Dimasp-D_n guardara los resultados de la etapa 1, 3 y 4, en archivos en disco duro. Por tal motivo en algunas de las siguientes gráficas se presenta el tiempo que tomaron los experimentos con y sin escritura de archivos en las etapas 3 y 4.

Los experimentos aquí mostrados fueron ejecutados sobre una laptop Centrino Duo a 1.6 Ghz con 1GB de RAM.

De manera similar a Dimasp-D_0, los experimentos se llevaron a cabo usando la colección de documentos Alex [ALEX], de la cual se escogieron los documentos *"Autobiography"* (de aproximadamente 100 páginas) *y "Letters"* (de aproximadamente 800 páginas) ambos del autor Thomas Jefferson. En ambos documentos no se eliminaron *stop-words*, sólo se omitieron números y símbolos de puntuación. Para los experimentos se tomaron del documento *"Autobiography"* las primeras 5000, 10000, 15000, 20000, 25000 y 30000 palabras. Las características de los subconjuntos de datos para *"Autobiography"* se muestran en la tabla 4.1. En el caso del documento *"Letters"* se tomaron incrementos de 40000 palabras, es decir se tomaron las primeras 40000, 80000, 120000, 160000, 200000, 240000 y 280000 palabras del documento. Las características de los subconjuntos de datos para *"Letters"* se muestran en la tabla 4.2.

Tabla 4.1 Características de los subconjuntos de datos usados para el documento *"Autobiography"*.

Número de palabras	**5,000**	**10,000**	**15,000**	**20,000**	**25,000**	**30,000**
Tamaño en KB	27	56	83	111	139	167
Caracteres procesados	22,599	46,328	69,861	92,868	116,578	140,193
Umbral β=0.04 % de las palabras en el doc.	2	4	6	8	10	12
Palabras diferentes	1,332	2,162	2,754	3,354	3,896	4,549
Secuencias de palabras diferentes con GAP=0	3,901	7,249	10,313	13,290	16,224	19,216
Secuencias de palabras diferentes con GAP=1	7,792	14,503	20,614	26,499	32,351	38,279
Secuencias de palabras diferentes con GAP=2	11,438	21,249	30,121	38,695	47,203	55,873

A continuación se presentan los experimentos realizados con Dimasp-D_n con el documento *"Autobiography"*; utilizando los diferentes subconjuntos de datos de la tabla 4.1. En la figura 4.30 se presenta la primera etapa variando el GAP desde 0 hasta 2.

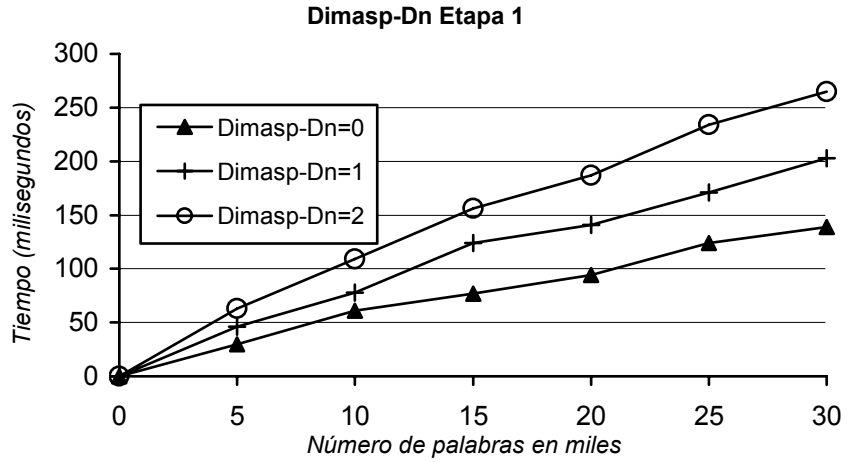


Figura 4.30 Tiempos obtenidos para la primera etapa de Dimasp-D_n.

La figura 4.31 muestra los tiempos requeridos para la construcción de la estructura de datos con el documento *"Autobiography"*.

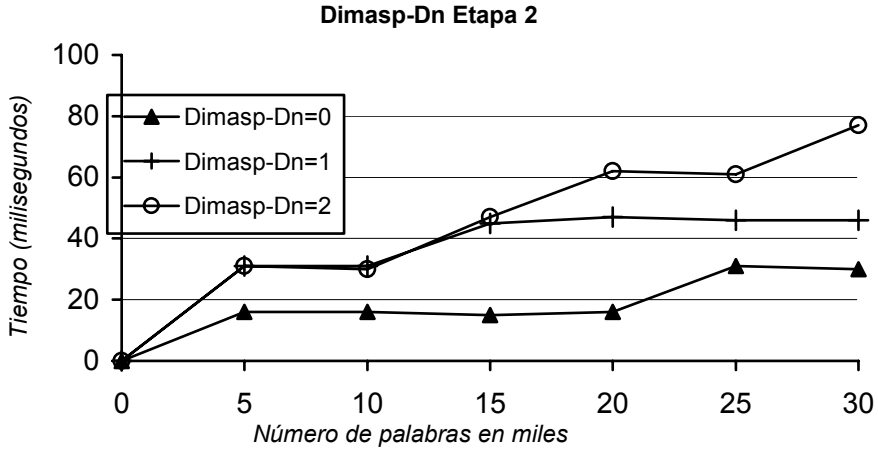


Figura 4.31 Tiempos obtenidos para la segunda etapa de Dimasp-D_n.

En la figura 4.32 se presentan los tiempos requeridos para la tercera etapa con el documento *"Autobiography"*, con y sin escritura de archivos temporales. Cabe señalar que para efectos de visualización en la gráfica de tiempos con escritura de archivos, se omitió el tiempo de procesamiento de 5000 palabras con GAP=2 que requirió 17 segundos.

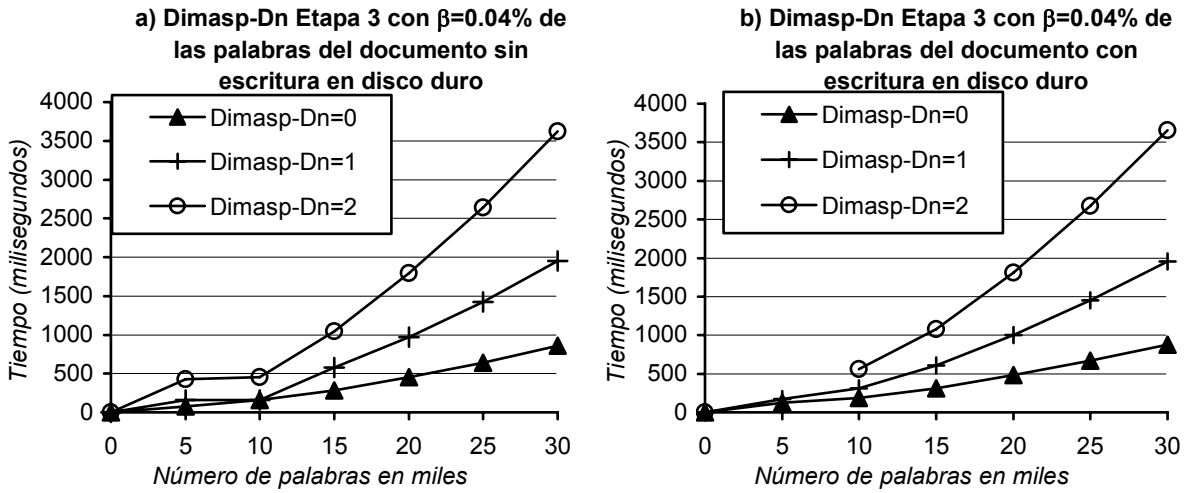


Figura 4.32 Tiempos obtenidos para la tercera etapa de Dimasp-C_n con y sin escritura de archivos temporales con $\beta = 0.04\%$ de la cantidad de palabras en el documento.

La figura 4.33 muestra los tiempos correspondientes a la cuarta etapa con el documento *"Autobiography"*, donde, para efectos de visualización, se omitieron los 37 y 104 segundos que requirieron para procesar 5000 palabras con GAP=2, con y sin escritura de archivos. En este sentido, se muestra que cuando existe un incremento en el tiempo de procesamiento, como es el caso con GAP=2 y 5000 palabras, existe un incremento en el tiempo de escritura en la misma etapa 3, lo cual hace que también afecte el tiempo de procesamiento en la cuarta etapa.

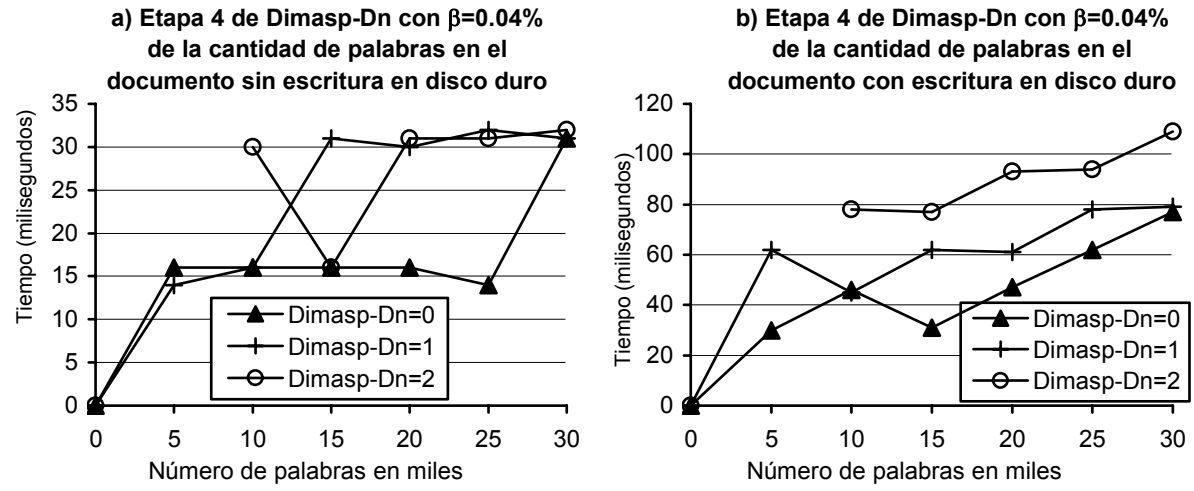


Figura 4.33 Tiempos obtenidos para la cuarta etapa de Dimasp-D_n con y sin escritura de archivos temporales con $\beta = 0.04\%$ de la cantidad de palabras en el documento.

En las gráficas de la figura 4.34 se muestran los tiempos de todas las etapas con y sin escritura de archivos. También en esta etapa se omitieron los 121 y 37 segundos requeridos para 5000 palabras con GAP=2, con y sin escritura de archivos temporales, respectivamente.

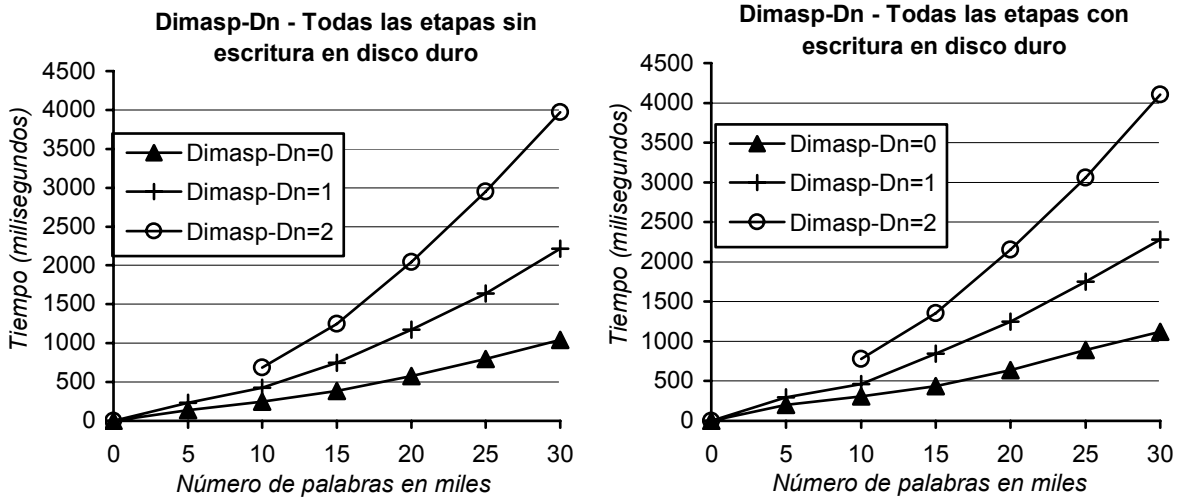


Figura 4.34 Tiempos obtenidos para todas las etapas de Dimasp-C_n con y sin escritura de archivos temporales con β = 0.04% de la cantidad de palabras en el documento.

Utilizando al documento *"Letters"* que es uno de los documentos más grandes de la colección Alex [ALEX] se realizaron pruebas similares al documento *"Autobiography"* pero ahora se fijó a β = 15 como el umbral para todos los experimentos. En este caso se utilizaron los subconjuntos de datos de la tabla 4.2.

Tabla 4.2 Características de los subconjuntos de datos usados para el documento *"Letters"*.

Número de palabras	**40,000**	**80,000**	**120,000**	**160,000**	**200,000**	**240,000**	**280,000**
Tamaño en KB	213	425	638	856	1,072	1,292	1,512
Caracteres procesados	177,297	354,324	533,269	715,549	897,666	1,082,082	1,267,270
Umbral β=15	15	15	15	15	15	15	15
Palabras diferentes	5,545	8,246	10,155	11,642	13,057	14,979	16,266
Pares de palabras diferentes con GAP=0	24,901	44,309	61,119	76,643	91,522	107,051	121,336
Pares de palabras diferentes con GAP=1	49,566	88,270	122,010	153,241	183,191	214,311	242,950
Pares de palabras diferentes con GAP=2	72,233	128,932	178,523	224,556	268,671	314,684	356,937

A continuación se presentan los experimentos realizados con Dimasp-D_n con el documento *"Letters"*; utilizando los diferentes subconjuntos de datos de la tabla 4.2. En la figura 4.35 se presenta la primera etapa variando el GAP desde 0 hasta 2. De acuerdo a esta gráfica es posible ver que la primera etapa presenta un comportamiento casi lineal con respecto al número de palabras en el documento.

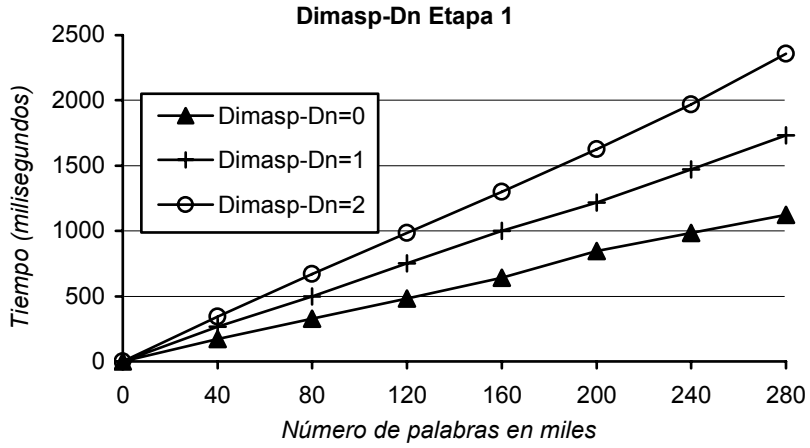


Figura 4.35 Tiempos obtenidos para la primera etapa de Dimasp-D_n con el documento *Letters*.

La figura 4.36 muestra el tiempo requerido para el documento *"Letters"* en la segunda etapa, variando el GAP desde 0 hasta 2. En esta gráfica es puede ver también que el tiempo de procesamiento se incrementa de manera lineal para diferentes subconjuntos del documento.

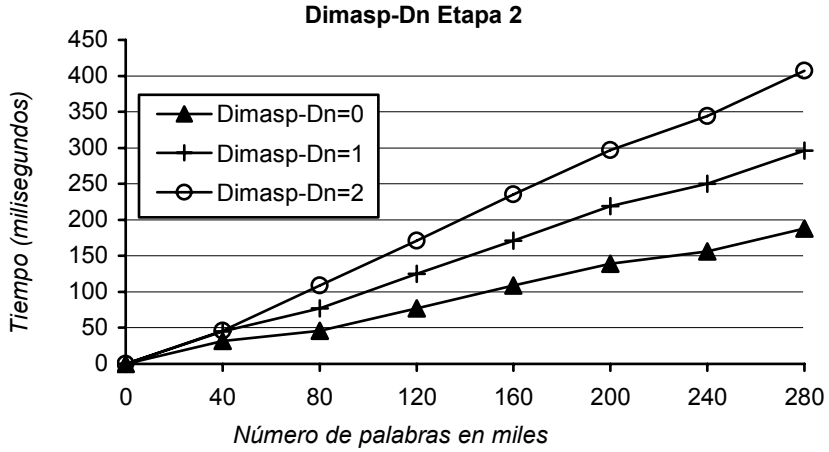


Figura 4.36 Tiempos obtenidos para la segunda etapa de Dimasp-D_n con el documento *Letters*.

En la figura 4.37 se presentan los tiempos requeridos en la tercera etapa con y sin escritura de archivos temporales. En esta gráfica se puede ver como afecta el tiempo de procesamiento cuando se incrementa el GAP.

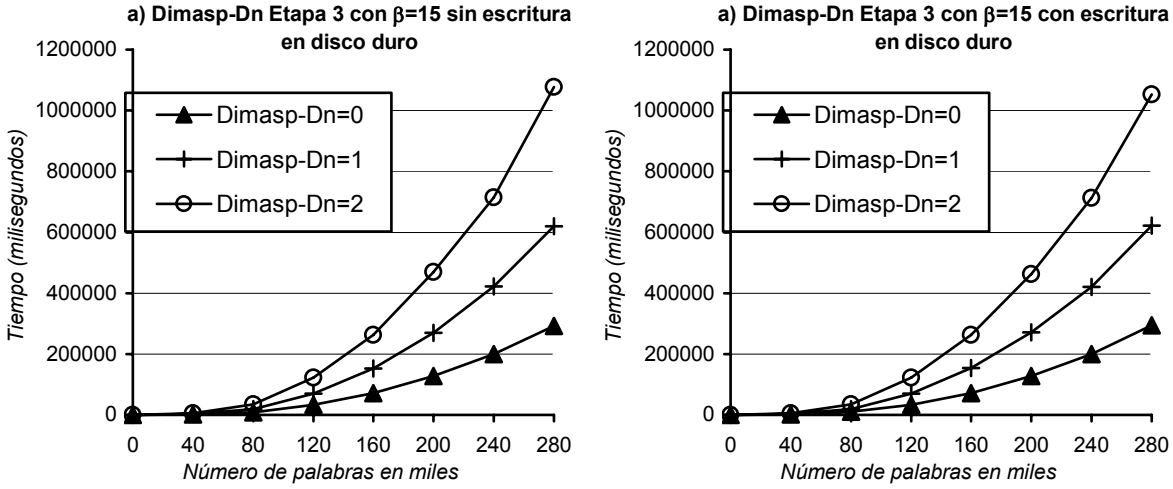


Figura 4.37 Tiempos obtenidos para la tercera etapa de Dimasp-D_n con y sin escritura de archivos temporales con $\beta = 15$.

En la figura 4.38 se presentan los tiempos requeridos por la cuarta etapa con y sin escritura de archivos temporales. En esta gráfica se puede ver como afecta el tiempo de procesamiento cuando se incrementa el GAP desde 0 hasta 2. Cabe señalar que todos los casos de la cuarta etapa se realizaron en menos de 600 milisegundos.

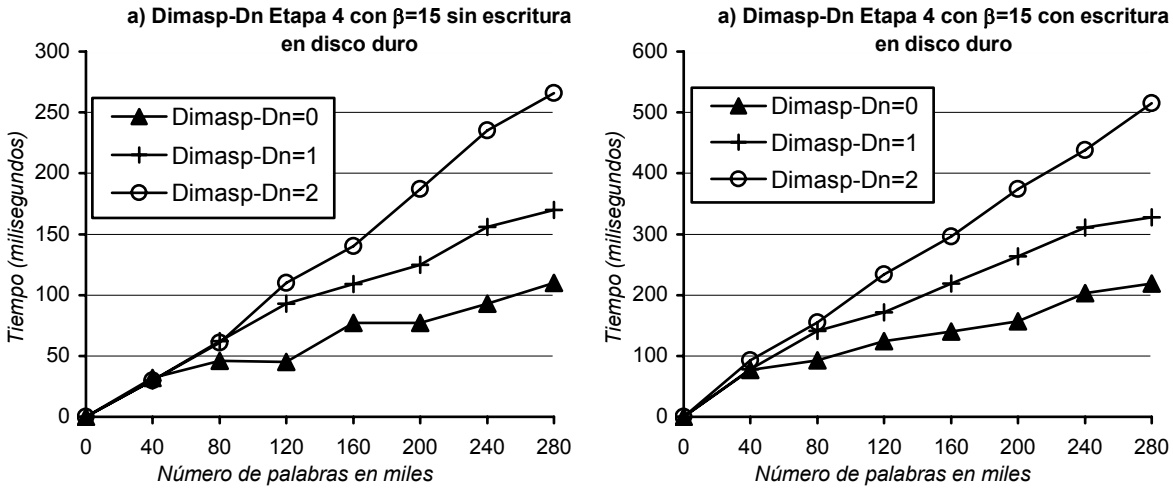


Figura 4.38 Tiempos de la cuarta etapa de Dimasp-D_n con y sin escritura de archivos temporales con $\beta = 15$.

En la figura 4.39 se muestran los tiempos totales requeridos para obtener las SFM's variando el GAP desde 0 hasta 2 con el documento *"Letters"*.

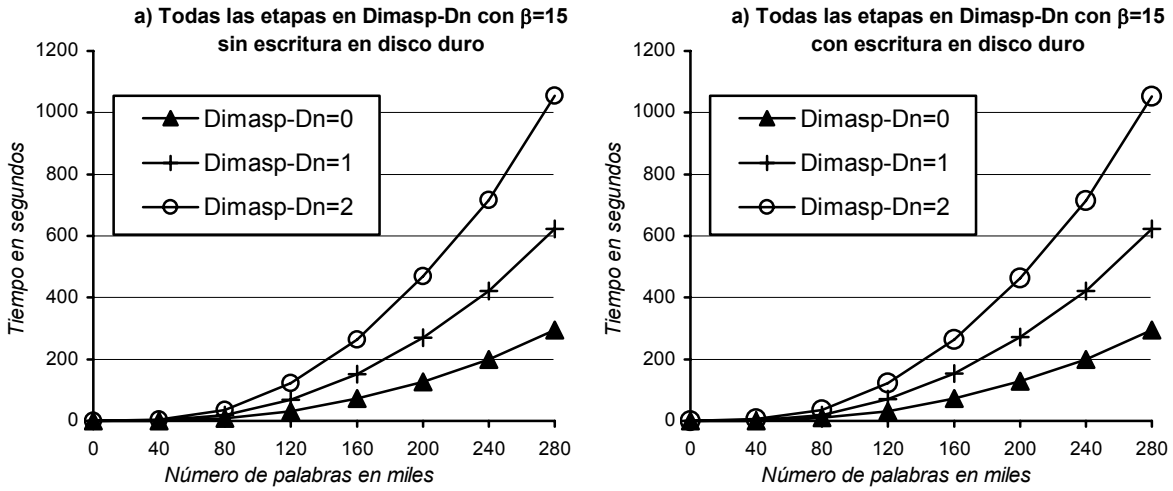


Figura 4.39 Tiempos obtenidos en la tercera etapa de Dimasp-D_n con y sin escritura de archivos temporales con $\beta = 15$.

Por último, con el objetivo de ver el aumento en la cantidad de SFM's cuando el GAP se incrementa, se muestra en la gráfica de la figura 4.40 el número de SFM's encontradas para GAP 0, 1 y 2; con β=15.

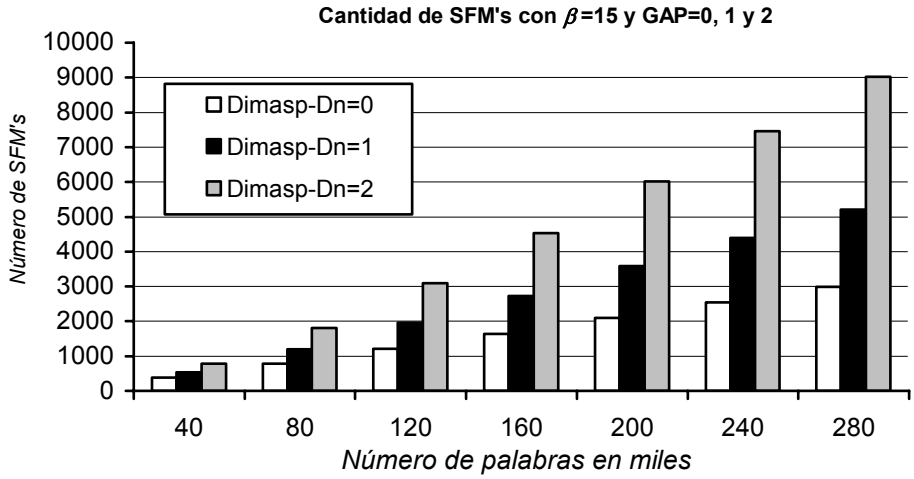


Figura 4.40 Número de SFM's con $\beta = 15$ para GAP 0, 1 y 2.

4.2.4 Sumario

En esta sección se planteó primero el problema de descubrimiento de SFM's en un solo documento con restricción GAP=n. Después, con base en la definición de este problema, se presentó a Dimasp-D_n un algoritmo de crecimiento de patrones para el descubrimiento de SFM's con GAP=n para un solo documento. Dimasp-D_n tiene la característica de poder obtener diferentes conjuntos de SFM's para diferentes umbrales de frecuencia sin la necesidad de reconstruir la estructura de datos, puesto que esta estructura no es modificada durante el proceso de búsqueda y es independiente del umbral requerido.

Dimasp-D_n encuentra todas las SFM's con GAP=n en un documento, pero por sus características este algoritmo puede actuar no exclusivamente sobre una secuencia de palabras sino sobre cualquier tipo de secuencia de datos.

CAPÍTULO 5

Conclusiones

En este capítulo se presentan las conclusiones generales del trabajo de investigación, así como las aportaciones principales. También se mencionan brevemente los trabajos en los que han sido usados los algoritmos de Dimasp-D_0 y Dimasp-C_0 para tareas en el procesamiento automático de textos. Por último, en la sección de trabajo futuro se mencionan algunas de las direcciones posibles de investigación a partir de esta tesis.

5.1. Conclusiones

En general, las estructuras de datos y las estrategias de búsqueda empleadas en Dimasp-C_0, Dimasp-C_n, Dimasp-D_0 y Dimasp-D_n, responde a la pregunta de investigación, puesto que fue posible desarrollar nuevos algoritmos que permiten hacer el descubrimiento de SFM's en

texto con restricción GAP (separación máxima que existe entre los elementos que forman una secuencia frecuente) de manera más rápida que los algoritmos antes propuestos.

Comúnmente, cuando se tienen n-objetos y se presenta el problema de búsqueda de un objeto en particular, el algoritmo más sencillo para la búsqueda es realizar una comparación para cada uno de los n-objetos, del que se está buscando. Sin embargo, para realizar este proceso más rápido, los algoritmos de búsqueda han desarrollado nuevas estructuras de datos a partir de los n-objetos de manera que, por un lado, se evite realizar las n-comparaciones, y por el otro lado, que las comparaciones que se tengan que realizar sean de manera rápida. Precisamente, los algoritmos y estructuras de datos que eviten el mayor número de comparaciones tendrán mejores tiempos de respuesta, sin olvidar que las estructuras de datos desarrolladas pueden crecer de manera exponencial. En este sentido, desde el inicio de la investigación se buscó construir una estructura de datos que no perdiera el orden secuencial que ya se tenía en las palabras, al contrario de algoritmos como GSP que incrementa el número de combinaciones de palabras a revisar por perder totalmente el orden secuencial que ya tenían las palabras. Esto nos permitió que los algoritmos desarrollados en esta tesis estén clasificados como de "crecimiento de patrones", los cuales en la práctica mejoran a los de "generación de secuencias candidatas"; lo cual sucede también en los experimentos de esta tesis.

En el desarrollo de los algoritmos de esta tesis, y en cada una de sus etapas, se buscó desarrollar estructuras de datos que tuvieran una construcción rápida y en un espacio lineal, pero que a la vez evitara en lo posible el mayor número de comparaciones. De hecho, se buscó que las estructuras de datos y estrategias de búsqueda desarrolladas en esta tesis fueran lo más sencillas posible puesto que esto afecta en el tiempo y espacio requerido para su construcción.

Cabe señalar, que los algoritmos como GSP, GenPrefixSpan y cSPADE son el resultado de adaptarle la restricción GAP a un algoritmo anterior. Sin embargo, los algoritmos desarrollados en esta tesis tenían que considerar la restricción GAP desde un inicio. Esto permitió buscar estructuras de datos y estrategias de búsqueda de manera más específica para este problema. Un ejemplo de ello es el árbol de la figura 3.41 de la tercera etapa el cual se construye de acuerdo al GAP, lo cual evita procesar un gran número de nodos redundantes.

En general, los algoritmos desarrollados trabajan en cuatro etapas, en la primera etapa se transforman los documentos a una representación basada en números enteros, lo cual hace

más rápido el manejo de los datos en etapas posteriores. Utilizando los documentos transformados, en la segunda etapa se construye una estructura de datos, a partir de la cual en la tercera etapa se encuentran las SF's que probablemente son maximales. Utilizando este conjunto de SF's, en la cuarta etapa se determinan cuáles son las SFM's. Cabe mencionar que las cuatro etapas fueron diseñadas para trabajar de manera separada por lo que es posible aprovechar todos los recursos de la computadora en cada etapa, permitiendo con ello procesar una mayor cantidad de datos.

En específico, en esta tesis se presentaron los algoritmos Dimasp-C_0 y Dimasp-C_n los cuales permiten descubrir SFM's considerando una colección de documentos con GAP=0 y GAP=n, respectivamente. De acuerdo a comparaciones experimentales, Dimasp-C_0 y Dimasp-C_n mejoran los tiempos de búsqueda de los algoritmos actuales (GSP, Delisp, GenPrefixSpan y cSPADE). En nuestros experimentos se muestra que Dimasp-C_0 mejora a Dimasp-C_n cuando GAP=0. Además, Dimasp-C_0 y Dimasp-C_n tienen la ventaja de poder reutilizar parte del trabajo realizado ya que la estructura de datos construida a partir de los documentos es independiente del umbral especificado por el usuario. Esta característica es especialmente útil en algunas aplicaciones donde es necesario probar con varios umbrales antes de encontrar uno apropiado. Otra de las ventajas que presenta Dimasp-C_0 y Dimasp-C_n es que pueden trabajar de manera incremental, para lo cual sólo es necesario procesar la parte correspondiente a los documentos recién agregados a la colección. Cabe señalar que esta característica sólo fue implementada en Dimasp-C_0, sin embargo su adaptación para Dimasp-C_n no sería complicada.

También se introdujo en esta tesis el problema de búsqueda de SFM's en un sólo documento el cual no había sido atacado antes y es importante pues no se pueden aplicar algoritmos como Dimasp-C. Para resolver este problema se introdujeron los algoritmos Dimasp-D_0 y Dimasp-D_n los cuales permiten descubrir SFM's considerando un sólo documento con GAP=0 y GAP=n, respectivamente. Para evaluar el desempeño de Dimasp-D_0 y Dimasp-D_n se utilizaron documentos de hasta 280,000 palabras (aproximadamente 800 páginas) con los cuales se obtuvo un buen desempeño.

Los experimentos presentados en esta tesis se realizaron sobre documentos de texto los cuales son descritos por secuencias de palabras. Sin embargo, las secuencias de palabras bien pudieran ser secuencias de ADN, secuencias de registros de visitas de paginas WEB, es decir, se puede procesar como entrada todo aquel objeto que describa, de manera categórica (mediante símbolos), un comportamiento secuencial.

5.2. Aportaciones

En los algoritmos desarrollados:

a) Un algoritmo para el descubrimiento de todas las SFM's en una colección de documentos con restricción GAP=0, Dimasp-C_0 y otro algoritmo para GAP=n, Dimasp-C_n; siendo GAP la máxima separación permitida entre los elementos que forman una secuencia frecuente. Los algoritmos Dimasp-C_0 y Dimasp-C_n han mostrado mejor comportamiento que otros algoritmos similares.

b) Un algoritmo para el descubrimiento de todas las SFM's en un solo documento con restricción GAP=0, Dimasp-D_0 y otro algoritmo para GAP=n, Dimap-D_n; siendo GAP la máxima separación permitida entre los elementos que forman una secuencia frecuente.

c) Una estrategia para utilizar estos algoritmos de manera incremental, la cual fue probada con Dimasp-C_0.

En aplicaciones de los algoritmos desarrollados:

d) Una biblioteca con las implementaciones de los cuatro algoritmos, de los cuales Dimasp-D_0 [García 2004] y Dimasp-C_0 [García 2006] han sido utilizados en diversos trabajos como:

e) Búsqueda de respuestas [Denicia 2006][Denicia 2007][Juárez 2007][Marina 2007].

f) Clasificación de documentos [Coyotl 2006][Coyotl 2007].

g) Generación de resúmenes [Villatoro 2006].

h) Agrupamiento de documentos [Hernández 2006][Hernández 2006a][Hernández 2007].

i) Construcción de catálogos de Hipónimos [Ortega 2007][Ortega 2007a].

j) Compresión de documentos [Osslan 2005]. En este último trabajo fue necesario que las apariciones de las SF's fueran mutuamente excluyentes. Es este caso se utilizó a

Dimasp-D_0 con uno de los mejores algoritmos de compresión, LZW. La idea básica del algoritmo LZW consiste en construir un diccionario de las cadenas de caracteres que se presentan en documento a comprimir. La compresión ocurre cuando son utilizados los índices en lugar de las cadenas de caracteres. Es decir la compresión ocurre cuando cadenas de caracteres frecuentes ya fueron leídas y son substituidas por un índice en el diccionario. En este trabajo se utilizó Dimasp-D_0 para buscar las SFM's de un documento las cuales son utilizadas como el diccionario inicial para el algoritmo de compresión LZW; consiguiendo mejorar la tasa de compresión.

5.3 Trabajo futuro

También se tiene pensado extender la capacidad de procesamiento incremental para los algoritmos Dimasp-C_n, Dimasp-D_0 y Dimasp-D_n.

Una de las pocas desventajas que pueden tener los algoritmos desarrollados es la necesidad de que la estructura de datos construida para Dimasp-C_0 y Dimasp-C_n debe caber en memoria RAM, lo cual podría evitarse si se utiliza la estrategia de bases de datos proyectadas de PrefixSpan y GenPrefixSpan para particionar la base de datos. Sin embargo, en nuestro caso sólo se harían las proyecciones necesarias hasta que la estructura de datos quepa en memoria RAM.

Para reducir aún más el tiempo de procesamiento en la tercera y cuarta etapa se tiene planeado procesar de manera paralela la estructura de datos. Esto puede realizarse teniendo la estructura de datos en común con varias computadoras las cuales obtendrían en la tercera etapa las SF's de una parte de la estructura de datos, luego cada equipo determinaría su conjunto de probables SFM's, las cuales tendrían que ser finalmente procesadas en un solo equipo.

Por otro lado, aunque ya se han utilizado las SFM's para tareas como compresión de datos [Osslan 2005], recuperación de información [Doucet 2004], respuesta a preguntas de definición [Denicia 2006], agrupamiento [Osslan 2005][Hernández 2006a] y clasificación de documentos [Coyotl 2006]; sería interesante utilizar las SFM's en otras tareas de procesamiento del lenguaje como lo son desambiguación del sentido de la palabra, resúmenes de documentos, extracción de información e incluso en el descubrimiento de

conocimiento en minería de texto o en la navegación de colecciones de documentos a partir de las SFM's.

Referencias

[Agrawal 1993] Agrawal Rakesh, Tomasz Imielinski Arun Swami. *"Mining Association Rules between Sets of Items in Large Databases"*. *Proc. ACM-SIGMOD* 1993 *Int'l Conference on Management of Data*, ISBN 0-89791-592-5, Washington D.C., May 1993, pp. 207–216.

[Agrawal 1994] Agrawal Rakesh, Srikant Ramakrishnan, *"Fast Algorithms for Mining Association Rules"*, Proceedings of 20th International Conference on Very Large Data Bases (VLDB), Morgan Kaufmann, ISBN 1-55860-153-8, 1994, pp. 487–499.

[Agrawal 1995] Agrawal R. and Srikant R. *"Mining Sequential Patterns"*, Proceedings of the 11th International Conference on Data Engineering, IEEE Computer Society, ISBN 0-8186-6910-1, Taiwan, March 1995, pp. 3–14.

[Agrawal 1996] Srikant R., Agrawal R. *"Mining sequential patterns: Generalizations and performance improvements"*. In 5th Intl. Conf. Extending Database Discovery and Data Mining, LNCS vol. 1057, ISBN 3-540-61057-X, March 1996, pp. 3–17.

[Ahonen 1997] Ahonen Helena, *et al, "Applying Data Mining Techniques in Text Analysis"*, Report C-1997-23, Department of Computer Science, University of Helsinki, 1997, pp. 1–12.

[Ahonen 1999] Ahonen Helena. *"Finding All Maximal Frequent Sequences in Text"*. Proceedings of the 16th International Conference on Machine Learning (ICML-99), D. Mladenic and M. Grobelnik (Eds.), Slovenia, June 1999, pp. 11–17.

[Ahonen 1999a] Ahonen Helena. *"Finding Co-occurring Text Phrases by Combining Sequence and Frequent Set Discovery"*, Proceedings of 16th

International Joint Conference on Artificial Intelligence (IJCAI-99), R. Feldman (Ed.), 1999, pp. 1–9.

[Ahonen 1999b] Ahonen Helena. *"Knowledge Discovery in Documents by Extracting Frequent Word Sequences"*. Library Trends on knowledge discovery in bibliographical databases, J. Qin and M.J. Norton (Eds.), 48(1), 1999, pp. 160–181.

[Ahonen 2002] Ahonen Helena. *"Discovery of Frequent Word Sequences in Text"*, Proceedings of ESF Exploratory Workshop on Pattern Detection and Discovery in Data Mining, LNCS vol. 2447, ISBN 3-540-44148-4, Springer Verlag, September 2002, pp. 180–189.

[Ahonen 2005] Ahonen-Myka H. and Doucet A. "*Data mining meets collocations discovery. In Inquiries into Words, Constraints, and Contexts".* A. Arppe *et al.* (Eds.) CSLI Studies in Computational Linguistics ONLINE. Ann Copestake (Series Editor), ISSN 1557-5772, CSLI Publications, Stanford, California, 2005, pp. 194–203.

[ALEX] Colección Alex. Documentos del dominio público sobre literatura y filosofía tanto inglesa como americana. http://www.infomotions.com/alex/.

[Antunes 2003] Antunes C., Oliveira A, *"Generalization of Pattern-growth Methods for Sequential Pattern Mining with Gap Constraints"*. 3rd Machine Learning and Data Mining in Pattern Recognition (MLDM 2003), LNCS vol. 2734, ISBN 978-3-540-40504-7, Springer Verlag, July 2003, pp. 239–251.

[Antunes 2004] Claudia Antunes, *"Sequential Pattern Mining Algorithms: trade-offs between speed and memory"*, in Workshop on Mining Graphs, Trees and Sequences (MGTS - ECML/PKDD 2004), Pisa, 2004.

[Antunes 2005] Claudia Antunes, *"Pattern Mining over Nominal Event Sequences using Constraint Relaxations"*, tesis de doctorado, Instituto Superior Técnico, Lisboa, January 2005, pp. 37–61.

[APRIORI] Christian-Borgelt, Implementación del algoritmo apriori http://www.borgelt.net/

[Arne 1999] Arne Andersson, N. Jesper Larsson, Kurt Swanson. *"Suffix Trees on Words"*. 7th Symposium on Combinatorial Pattern Matching, 1999.

[Bamshad 2002] Bamshad Mobasher, Honghua Dai, Tao Luo, Miki Nakagawa, *"Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks"*, In Proceedings of the IEEE International Conference on Data Mining (ICDM'02), IEEE Computer Society, ISBN 0-7695-1754-4, Maebashi City, Japan, 2002, p. 669.

[Burdick 2001] Burdick Doug. *"MAFIA: A Maximal Frequent Itemset Algorithms for Transactional Databases"*, Proceedings of the 17th International Conference on Data Engineering, IEEE Computer Society, ISBN 0-7695-1001-9, 2001, pp. 443–452.

[Coyotl 2006] Coyotl-Morales Rosa Maria, Villaseñor-Pineda Luis, Montes y Gómez Manuel, Rosso Paolo. *"Authorship Attribution Using Word Sequences*". 11th Iberoamerican Congress on Pattern Recognition (CIARP'2006), LNCS vol. 4225, ISBN 3-540-46556-1, Springer Verlag, 2006, pp. 844–853.

[Coyotl 2007] Rosa María Coyotl Morales, *"Clasificación Automática de Textos considerando el Estilo de Redacción"*, Tesis de Maestría, Instituto Nacional de Astrofísica, Óptica y Electrónica, 2007.

[Dao 1998] Lin Dao-l. *"Fast Algorithms for Discovering the Maximum Frequent Set"*, Tesis Doctoral, New York University, 1998.

[Dao 2002] Dao-l Lin, Zvi M. Kedem, *"Pincer-Search: An Efficient Algorithm for Discovering the Maximum Frequent Set"*, IEEE Transactions on Knowledge and Data Engineering, vol. 14, issue 3, 2002, pp. 553–566.

[Denicia 2006] Claudia Denicia-Carral, Manuel Montes-y-Gómez, Luis Villaseñor-Pineda, René García Hernández. "*A Text Mining Approach for Definition Question Answering"*, 5th International Conference on NLP (Fintal 2006), ISBN 978-3-540-37334-6, LNAI vol. 4139, Springer-Verlag 2006, pp. 76–86.

[Denicia 2007] María Claudia Denicia Carral, *"Respondiendo Preguntas de Definición mediante el Descubrimiento de Patrones Léxicos"*. Tesis de Maestría, Instituto Nacional de Astrofísica, Óptica y Electrónica, 2007.

[Doucet 2003] Antoine Doucet and Helena Ahonen-Myka, "*Naive clustering of a large XML document collection*". Proceedings of the 1st Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX), ERCIM Workshop Proceedings, 2003, pp. 81–88.

[Doucet 2004] Doucet Antoine, Ahonen Helena, *"Non-contiguous Word Sequences for Information Retrieval", Proceedings of the 42nd annual meeting of the Association for Computational Linguistics (ACL-2004)*, 2004, Workshop on Multiword Expressions: Integrating Processing, Spain, July, 2004, pp. 88–95.

[Doucet 2004a] Doucet Antoine, *"Utilisation de Séquences Fréquentes Maximales en Recherche d'Information"*, Proceedings of the 7th International Conference on the Statistical Analysis of Textual Data (JADT 2004), Belguim, March 2004, pp. 334–345.

[Doucet 2004b] Doucet Antoine, *et al., "Accurate Retrieval of XML Document Fragments using EXTIRP"*, Proceedings of the Second Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX), Germany, December 2004, pp. 73–80.

[Fayyad 1996] Fayyad U., Piatetsky-Shapiro G. *"From Data Mining to Knowledge Discovery in Databases"*, Proceedings of the 1st International Conference on Knowledge Discovery, 1996, pp. 37–54.

[FIMI 2003] FIMI'03: Workshop on Frequent Itemset Mining Implementations http://FIMI.CS.HELSINKI.FI/FIMI03/

[FIMI 2004] FIMI'04: Workshop on Frequent Itemset Mining Implementations http://FIMI.CS.HELSINKI.FI/FIMI04/

[Feldman 1995] Feldman R. and Dagan I. "Knowledge Discovery in Textual Databases (KDT)", Proceedings of the 1st International Conference on Knowledge Discovery (KDD-95), Montreal, August 1995, pp. 112–117.

[Feldman 1996] Feldman R. and Hirsh Hyam. "Mining Associations in Text in the Presence of Background Knowledge", Proceedings of the 2nd International

Conference on Knowledge Discovery from Databases, Portland, August 1996, pp. 343–346.

[Frank 1999] Frank Eibe, Paynter Gordon, Witten Ian. *"Domain-Specific Keyphrase Extraction"*, Proceedings of the 16th International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers, ISBN 1-55860-613-0, 1999, pp. 668-673.

[García 2004] René A. García-Hernández, José Fco. Martínez-Trinidad and Jesús Ariel Carrasco-Ochoa, "*A Fast Algorithm to Find All the Maximal Frequent Sequences in a Text*", 9th Iberoamerican Congress on Pattern Recognition (CIARP'2004), LNCS vol. 3287, Springer-Verlag 2004, pp. 478–486.

[García 2006] René A. García-Hernández, José Fco. Martínez-Trinidad and Jesús Ariel Carrasco-Ochoa, "*A New Algorithm for Fast Discovery of Maximal Sequential Patterns in a Document Collection*", 7th Intelligent Text Processing and Computational Linguistics (CICLing'2006), LNCS vol. 3878, Springer-Verlag 2006, pp. 514-523.

[Hajime 2002] Hajime Kitakami, Tomoki Kanbara, et al., *"Modified PrefixSpan Method for Motif Discovery in Sequence Databases"*, PRICAI 2002: Trends in Artificial Intelligence: 7th Pacific Rim International Conference on Artificial Intelligence, ISSN 0302-9743, LNCS vol. 2417, Springer-Verlag, 2002, pp. 311–318.

[Han 2000] Han J, Pei J, *"FreeSpan: Frequent Pattern-Projected Sequential Patterns Mining"*, International Conference Knowledge Discovery in Databases (KDD'00), ISBN 1-58113-233-6, Boston, USA, 2000, pp. 355–359.

[Han 2000a] Han Jiawei, et al, *"Mining Frequent Patterns without Candidate Generation"*, Proceedings International Conference on Management of Data (SIGMOD'00), Dallas, USA, May 2000, pp. 1–12.

[HIMALAYA] Himalaya Data Mining Tools, http://himalaya-tools.sourceforge.net/.

[Hernández 2006] Hernández-Reyes Edith, Carrasco-Ochoa J. A., Martínez-Trinidad J. Fco, García-Hernández Rene A. *"Document Representation Based on*

Maximal Frequent Sequence Sets". 11th Iberoamerican Congress on Pattern Recognition (CIARP 2006), ISBN 3-540-46556-1, LNCS vol. 4225, Springer-Verlag, Cancun, Mexico, 2006, pp. 854–863.

[Hernández 2006a] Hernández-Reyes Edith, García-Hernández Rene A., Carrasco-Ochoa J. A., Martínez-Trinidad J. Fco. "*Document Clustering based on Maximal Frequent Sequences"*, 5th International Conference on NLP (Fintal 2006), Tapio Salakoski et al. (Eds.), ISBN 3-540-37334-9, LNCS vol. 4139, Springer-Verlag, Turku, Finland, August 2006, pp. 257-267.

[Hernández 2007] Edith Hernández Reyes, *"Agrupamiento de documentos basado en Secuencias Frecuentes Maximales".* Tesis de Maestría, Instituto Nacional de Astrofísica, Óptica y Electrónica, 2007.

[Hunor 2003] Hunor Albert-Lorincz, J-F. Boulicaut. "*A framework for frequent sequence mining under generalized regular expression constraints"*. Proceedings of the 2nd International Workshop on Knowledge Discovery in Inductive Databases KDID'03 co-located with ECML-PKDD 2003, J-F. Boulicaut and S. Dzeroski (Eds.), Cavtat-Dubrovnik (Croatia), ISBN 953-6690-34-9, September 22, 2003, pp. 2-16.

[Jay 2002] Jay Ayres, Johannes Gehrke, Tomi Yiu, Jason Flannick. *"Sequential PAttern Mining using a Bitmap Representation"*, Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, ISBN 1-58113-567-X, Canada, 2002, pp. 429–435.

[Jeong 2005] Jeong Hee Hwang and Keun Ho Ryu, *"A New Sequential Mining Approach to XML Document Clustering"*, Web Technologies Research and Development - APWeb 2005, ISBN 3-540-25207-X, LNCS 3399, Springer-Verlag, March 2005, pp. 266–276.

[Jeong 2005a] Jeong Hee Hwang and Keun Ho Ryu, *"Clustering and Retrieval of XML Documents by Structure"*, Computational Science and Its Applications – ICCSA 2005, ISBN 3-540-25861-2, LNCS 3481, Springer-Verlag, 2005, pp. 925–935.

[Juárez 2007] Antonio Juárez-González, Alberto Téllez-Valero, Claudia Denicia-Carral, Manuel Montes-y-Gómez and Luis Villaseñor-Pineda, *"Using Machine Learning and Text Mining in Question Answering"*, 7th Workshop of the Cross-Language Evaluation Forum (CLEF 2007), LNCS 4730, Springer 2007.

[Karanikas 2002] Karanikas Haralampos, Theodoulidis Babis, "*Knowledge Discovery in Text and Text Mining Software"*, Technical Report, Centre for Research in Information Management (CRIM), Department of Computation, UMIST, 2002. http://www.crim.co.umist.ac.uk/parmenides/

[Kodratoff 1999] Kodratoff Yves. *"Knowledge Discovery in Texts: A Definition and Applications"*, Proceedings of the 11th International Symposium on Foundations of Intelligent Systems, Ras & Skowron (Eds.), ISBN 3-540-65965-X, LNAI 1609, Springer, 1999, pp. 16–29.

[Leavitt 2002] Leavitt Neal, "Data Mining for the Corporate Masses", IEEE Computer Society Press, ISSN 0018-9162, vol. 35, Issue 5, May 2002, pp. 22–24.

[Lent 1997] Lent Brian, Agrawal Rakesh, Ramakrishnan Srikant, *"Discovering Trends in Text Databases"*, Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, California, USA, 1997, pp. 227–230.

[LEWIS] Colección Reuters-21578 de la agencia Reuters con 21578 noticias, http://www.daviddlewis.com/resources/testcollections/ reuters21578/.

[Lorincz 2003] H. Albert-Lorincz, J-F. Boulicaut. *"Mining frequent sequential patterns under regular expressions: a highly adaptive strategy for pushing constraints"*. Proceedings of the 3rd SIAM International Conference on Data Mining SDM'03, D. Barbara and C. Kamath (Eds.), San Francisco, USA, May, 2003, pp. 316–320.

[Marina 2007] Rita Marina Aceves-Pérez, Manuel Montes-y-Gómez, Luis Villaseñor Pineda, *"Enhancing Cross-Language Question Answering by Combining Multiple Question Translations"*, 8th Intelligent Text Processing and Computational Linguistics (CICLing'2007), LNCS 4394, Springer-Verlag 2007, pp. 485–493.

[Ming 2002] Ming-Yen Lin, Suh-Yin Lee, and Sheng-Shun Wang, *"DELISP: Efficient Discovery of Generalized Sequential Patterns by Delimited Pattern-Growth Technology,"* Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD02), Taipei, Taiwan, 2002, pp. 189–209.

[Ming 2002a] Ming-Yen Lin and Suh-Yin Lee, *"Fast Discovery of Sequential Patterns by Memory Indexing"*, Data Warehousing and Knowledge Discovery: 4th International Conference, DaWaK. Proceedings, LNCS 2454, Springer-Verlag, France, September 2002, pp. 150–160.

[Ming 2003] Ming-Yen Lin, *"Efficient Algorithms for Association Rule Mining and Sequential Pattern Mining"*, Tesis doctoral, College of Electrical Engineering and Computer Science, National Chiao Tung University, China, 2003.

[Minos 1999] Minos N. Garofalakis, Rajeev Rastogi, Kyuseok Shim, *"SPIRIT: sequential pattern mining with regular expression constraints"*, International Conference on Very Large Databases, ISBN 1-55860-615-7, Morgan Kaufmann, 1999, pp. 223–234.

[Mohammed 2000] Mohammed Zaki, "SPADE: An Efficient Algorithm for Mining Frequent Sequences", Machine Learning, Kluwer Academic Publishers, vol. 42, 2000, pp. 31–60.

[Mohammed 2000a] Mohammed J. Zaki, *"Sequence Mining in Categorical Domains: Incorporating Constraints"*, in 9th International Conference on Information and Knowledge Management, Washington, November 2000, pp. 422–429.

[Montes 2002] Manuel Montes y Gómez, *"Minería de texto empleando la semejanza entre estructuras semánticas"*, Tesis de doctorado, Instituto Politécnico Nacional, 2002.

[Orlando 2004] S. Orlando, R. Perego, C. Silvestri. *"A new algorithm for gap constrained sequence mining"*. Proceedings of ACM Symposim on Applied Computing (SAC), special track on Data Mining, Cyprus, March 2004, pp. 540–547.

[Ortega 2007] Rosa M. Ortega-Mendoza, Luis Villaseñor Pineda, Manuel Montes-y-Gómez. "*Construcción Automática de Catálogos de Hipónimos a partir de Texto no estructurado*". E2C2 2007: Memorias de Investigación en Computación del Instituto Politécnico Nacional, México 2007.

[Ortega 2007a] Rosa M. Ortega-Mendoza, Luis Villaseñor-Pineda, Manuel Montes-y-Gómez, *Using Lexical Patterns For Extracting Hyponyms from the Web*. 6th Mexican International Conference on Artificial Intelligence (MICAI 2007), LNAI, Springer-Verlag, 2007. (por aparecer)

[Osslan 2005] Osslan O. Vergara-Villegas, René A. García-Hernández, J. Ariel Carrasco-Ochoa, Raúl Pinto Elías, José F. Martínez-Trinidad. "*Data Preprocessing by Sequential Pattern Mining for LZW*", 6th Mexican International Conference on Computer Science (ENC 2005), ISBN 0-7695-2454-0, IEEE Computer Society Press, September 2005. pp. 82–87.

[Pei 2001] Pei J, Han, *et al*, "*PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth*", *Proceedings of 17th International Conference on Data Engineering* (ICDE 01), 2001, pp. 215–226.

[Pei 2001a] Pei-Hsin Wu, Wen-Chih Peng, Ming-Syan Chen, "*Mining Sequential Alarm Patterns in a Telecommunication Database*", Proceedings of the VLDB-2001 International Workshop on Databases in Telecommunications II, ISBN 3-540-42623-X, LNCS 2209, Springer-Verlag, 2001, pp. 37–51.

[Pei 2002] Pei Jian, "*Pattern-Growth Methods for Frequent Pattern Mining*", Tesis Doctoral, Simon Fraser University, Canada, 2002.

[Pei 2002a] Pei Jian, Han Jiawei, Wang Wei, "*Mining Sequential Patterns with Constraints in Large Databases*", Proc. of International Conference on Information and Knowledge Management (CIKM'02), 2002, pp. 18–25.

[Pei 2004] Pei Jian, Han Jiawei, et al., "*Mining Sequential Patterns by Patterns-Growth: The PrefixSpan Approach*", IEEE Transactions on Knowledge and Data Engineering, November 2004, vol. 16, num. 11, pp. 1424–1440.

[Perego 2003] R. Perego, C. Silvestri *"CCSM: an Efficient Algorithm for Constrained Sequence Mining. S. Orlando"*. Proceedings of the 6th International Workshop on High Performance Data Mining, in conjunction with 3rd International SIAM Conference on Data Mining (HPDM), San Francisco, May 2003.

[QUEST] IBM, Generador de bases de secuencias sintéticas http://www.almaden.ibm.com/software/quest/Resources/ index.shtml.

[Seno 2002] Seno Masakazu, *"SLPMiner: An Algorithm for Finding Frequent Sequential Patterns Using Length-Decreasing Support Constraint", IEEE International Conference on Data Mining*, IEEE Computer society, ISBN 0-7695-1754-4, 2002, p. 418.

[Sholom 2005] Sholom M. Weiss, Mitin Indurkhya, Tong Damerau, *"Text mining: Predictive Methods for Analyzing Unstructured Information"*. ISBN-10 0387954333, Springer-Verlag, 2005.

[Sujeevan 2005] Aseervatham Sujeevan and Osmani Aomar, *"Mining short sequential patterns for hepatitis type detection"*, European Conferences on Machine Learning and European Conferences on Principles and Practice of Knowledge Discovery in Databases, Portugal, 2005.

[Turney 1997] Turney P. *"Extraction of keyphrases from Text: Evaluation of Four Algorithms"*, National Research Council of Canada, Technical Report ERB-1051, 1997.

[Turney 1999] Turney P, *"Learning Algorithms for Keyphrase Extraction"*, National Research Council of Canada, NRC-44105, Information Retrieval, 1999, pp. 303–336.

[Turney 1999a] Turney P, *"Learning to Extract Keyphrases from Text"*, National Research Council of Canada, Technical Report ERB-1057, 1999.

[Villatoro 2006] Esaú Villatoro Tello, *"Generación Automática de Resúmenes de Múltiples Documentos"*, Tesis de Maestría, Instituto Nacional de Astrofísica Óptica y Electrónica, 2006.

[Yang 2004] Yang Guizhen, *"The Complexity of Mining Maximal Frequent Itemsets and Maximal Frequent Patterns"*, Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD2004), ACM Press, ISBN 1-58113-888-1, USA, 2004, pp. 344–353.

[Yap 2006] Ivan Yap, Han Tongh Loh, et al, *"Topic Detection Using MFSs"*, 19th International Conference and Other Applications of Applied Intelligent Systems, IEA/AIE, ISBN 3-540-35453-0, LNCS 4031, Springer Verlag, France, 2006, pp. 342–352.

[Youssefi 2004] Amir H. Youssefi, David J. Duke, Mohammed J. Zaki, Ephraim P. Glinert, *"Visual Web Mining"*, 13th International World Wide Web Conference, ACM, ISBN 1-58113-912-8, New York, 2004, pp. 394–395.

Anexo 1.

Lista de palabras eliminadas en la colección reuters-21578

1. A
2. ABOUT
3. ADDED
4. AFTER
5. AGAINST
6. AGREED
7. AGREEMENT
8. ALL
9. ALSO
10. AN
11. AND
12. ANNOUNCED
13. ANNUAL
14. ANY
15. APRIL
16. ARE
17. AT
18. AVG
19. B
20. BANK
21. BANKS
22. BASED
23. BE
24. BECAUSE
25. BEEN
26. BEFORE
27. BEING
28. BETWEEN
29. BILLION
30. BOARD
31. BOTH
32. BUSINESS
33. BUT
34. BUY
35. BY
36. C
37. CAN
38. CASH
39. CO
40. COMMON
41. CORP
42. COULD
43. CREDIT
44. CTS
45. CURRENT
46. CUT
47. DAY
48. DEBT
49. DID
50. DIV
51. DLR
52. DLRS
53. DO
54. DOLLAR
55. DOWN
56. DUE
57. DURING
58. EACH
59. EARLIER
60. END
61. EXCHANGE
62. EXPECTED
63. FEB
64. FEBRUARY
65. FIRST
66. FIVE
67. FOR
68. FOREIGN
69. FOUR
70. FROM
71. FURTHER
72. GENERAL
73. GROUP
74. H
75. HAD
76. HAS
77. HAVE
78. HE
79. HIGH
80. HIGHER
81. HIS
82. HOWEVER
83. IF
84. IN
85. INC
86. INCLUDING
87. INTEREST
88. INTO
89. INVESTMENT
90. IS
91. ISSUE
92. IT
93. ITS
94. J
95. JANUARY
96. JUNE
97. LAST
98. LONG
99. LOSS
100. LOWER
101. LT
102. LTD
103. MADE
104. MAJOR
105. MAKE
106. MANAGEMENT
107. MARCH
108. MAY
109. MLN
110. MONTH
111. MONTHS
112. MORE
113. MOST
114. MTHS
115. N
116. NET
117. NEW
118. NEXT
119. NINE
120. NO
121. NOT
122. NOTE
123. NOW
124. OF
125. OFFER
126. OIL
127. ON
128. ONE
129. ONLY
130. OR
131. OTHER
132. OUT
133. OVER
134. P
135. PART
136. PAY
137. PCT
138. PER
139. PERIOD
140. PLANS
141. PRICE
142. PRICES
143. PROFIT
144. PUBLIC
145. QTLY
146. QTR
147. QUARTER
148. RATE
149. RATES
150. RD
151. RECORD
152. REPORTED
153. REUTER
154. REVS
155. RISE
156. ROSE
157. S
158. SAID
159. SALE
160. SALES
161. SAME
162. SAYS
163. SECURITIES
164. SET
165. SEVEN
166. SHARE
167. SHARES
168. SHOULD
169. SHR
170. SHRS
171. SINCE
172. SIX
173. SO
174. SOME
175. SOURCES
176. SPOKESMAN
177. ST
178. STATE
179. STATEMENT
180. STOCK
181. SUCH
182. T
183. TAKE
184. TAX
185. TERM
186. TERMS
187. TH
188. THAN
189. THAT
190. THE
191. THEIR
192. THERE
193. THEY
194. THIS
195. THREE
196. THROUGH
197. TIME
198. TO
199. TODAY
200. TOLD
201. TOTAL
202. TRADE
203. TRADING
204. TWO
205. U
206. UNDER
207. UNIT
208. UP
209. VS
210. WAS
211. WE
212. WEEK
213. WERE
214. WHEN
215. WHICH
216. WHILE
217. WHO
218. WILL
219. WITH
220. WITH
221. WORLD
222. WOULD
223. Y
224. YEAR
225. YEARS
226. YESTERDAY

Anexo 2.

Algunas noticias de la colección reuters-21578 donde se encontró alguna SFM

Ejemplo 1

Utilizando el algoritmo Dimasp-C_0 con β=15 se encontró una SFM de logitud 25, la cual se marca en negritas en la noticia 82421. Cabe señalar que la SFM encontrada es la más grande.

Noticia 82421
HOG AND **CATTLE SLAUGHTER GUESSTIMATES**

Chicago Mercantile Exchange **floor traders** and **commission house representatives** are **guesstimating** today's **hog slaughter** at about 295,000 to 305,000 **head versus** 295,000 week **ago** and 305,000 a year **ago.**

Cattle slaughter is **guesstimated** at about 120,000 to 125,000 **head versus** 114,000 week **ago** and 119,000 a year **ago**.

Ejemplo 2

Utilizando el algoritmo Dimasp-C0 con β=3 se encontró una SFM de logitud 115, la cual se marca en negritas en las noticias que se encontró. Cabe señalar que la SFM encontrada es la más grande.

Noticia: 10268 Titulo: "INDIA STEPS UP COUNTERTRADE DEALS TO CUT TRADE GAP"

India is **searching** for **non-communist countertrade partners** to **help** it cut its trade **deficit** and **conserve** foreign exchange.

Wheat, **tobacco, tea, coffee, jute, engineering** and **electronic goods, as well as minerals** including **iron ore**, are all on offer in **return** for **crude** oil, **petroleum products, chemicals, steel** and **machinery**, trade sources told Reuters.

Most of the **impetus behind countertrade**, which **began** in 1984, **comes** from two state trading **firms** -- the State Trading Corp (**STC**) and the **Minerals** and **Metals** Trading Corp (**MMTC**).

"The two state trading **corporations** are **free** to **use** their **buying power** in **respect** to **bulk commodities** to **promote Indian exports**," a **commerce ministry spokeswoman** said, **adding** that **private firms** are **excluded** from **countertrading**.

One trade **source** said **India** has **targetted countries** that **depend** on an **Indian domestic market recently opened** to foreign **imports**.

However, **countertrade deals still** make up only a **small** part of **India**'s total trading and are **likely** to **account** for **less** than **eight** pct of the **estimated** 18.53 billion dlrs in trade
during the nine months **ended December**, the sources said.

Countertrade accounted for **just** five pct of **India**'s 25.65 billion dlrs in trade during **fiscal** 1985/86 **ended** March, against **almost nothing** in 1984/85, **official figures show**.

However, the **figures exclude exchanges** with the **Eastern Bloc paid** in **non-convertible Indian rupees**, the sources said.

Total trade with the **Soviet Union**, **involving swaps** of **agricultural produce** and **textiles** for **Soviet arms** and **crude** oil, is **estimated** at 3.04 billion dlrs in **fiscal** 1986/87, against three billion in 1985/86.

Indian countertrade, which is being promoted mainly to help narrow the country's large trade deficit, is still insignificant compared with agreements reached by Indonesia, Venezuela and Brazil, the trade sources said.

The trade deficit, which hit an estimated record 6.96 billion dlrs in 1985/86, is expected to decline to 5.6 billion in the current fiscal year.

But the push to include non-communist countries in countertrade is also due to other factors, including the slow growth of foreign reserves, a tight debt repayment schedule, shrinking aid and trade protectionism, businessmen said.

One source said India is showing more dynamism in promoting countertrade deals than in the past, when the deals were made discreetly because they break GATT rules. As a member of the General Agreement on Tariffs and Trade (GATT), India cannot officially support bartering.

The MMTC's recent countertrade deals include iron ore exports to Yugoslavia for steel structures and rails.

"MMTC's recent global tenders now include a clause that preference will be given to parties who accept payment in kind for goods and services sold to India," a trade official said,
adding that the policy remains flexible.

"We also take into account other factors such as prices at which the goods and services are offered to India," the trade official said.

Early this year the commerce ministry quietly told foreign companies interested in selling aircraft, ships, drilling rigs and railway equipment to India that they stood a better chance if they bought Indian goods or services in return, the trade sources said.

Illustrating the point, the official said a South Korean firm recently agreed to sell a drilling platform worth 40 mln dlrs to the state-run Oil and Natural Gas Commission. In return, the South Koreans gave a verbal assurance to buy Indian goods worth 10 pct of the contract, against the 25 pct sought by New Delhi, the trade official said.

"We selected the Korean firm because its bid was the lowest," he added.

Countertrade is helping African countries short of foreign currency to import goods. India has signed a trade protocol to buy up to 15,000 tonnes of asbestos fibre from Zimbabwe in exchange for Indian goods, including jute bags and cars. But despite India's new drive, countertrade has some inherent problems, they added.

"It is not always easy to meet the basic requirement that the trade should always be balanced," one trade source said. "The other problem is it is often difficult to supply or buy commodities which the other party wants."

Another added, "Barter is also restrictive. We look upon it as a temporary measure to get over the current balance of payments difficulty.

"This is why countertrade has not been made a law in India.
It does not even figure in the country's foreign trade policy."

Noticia: 10375 Título: “INDIA STEPS UP COUNTERTRADE DEALS TO CUT TRADE GAP”

India is **searching** for **non-communist countertrade partners** to **help** it cut its trade **deficit** and **conserve** foreign exchange.

Wheat, **tobacco, tea, coffee, jute, engineering** and **electronic goods, as well as minerals** including **iron ore**, are all on offer in **return** for **crude** oil, **petroleum products, chemicals, steel** and **machinery**, trade sources told Reuters.

Most of the **impetus behind countertrade**, which **began** in 1984, **comes** from two state trading **firms** -- the State Trading Corp (**STC**) and the **Minerals** and **Metals** Trading Corp (**MMTC**).

"The two state trading **corporations** are **free** to **use** their **buying power** in **respect** to **bulk commodities** to **promote Indian exports**," a **commerce ministry spokeswoman** said, **adding** that **private firms** are **excluded** from **countertrading**.

One trade **source** said **India** has **targetted countries** that **depend** on an **Indian domestic market recently opened** to foreign **imports**.

However, **countertrade deals still** make up only a **small** part of **India**'s total trading and are **likely** to **account** for **less** than **eight** pct of the **estimated** 18.53 billion dlrs in trade
during the nine months **ended December**, the sources said.

Countertrade accounted for **just** five pct of **India**'s 25.65 billion dlrs in trade during **fiscal** 1985/86 **ended** March, against **almost nothing** in 1984/85, **official figures show**.

However, the **figures exclude exchanges** with the **Eastern Bloc paid** in **non-convertible Indian rupees**, the sources said.

Total trade with the **Soviet Union**, **involving swaps** of **agricultural produce** and **textiles** for **Soviet arms** and **crude** oil, is **estimated** at 3.04 billion dlrs in **fiscal** 1986/87, against three billion in 1985/86.

Indian countertrade, which is being promoted mainly to help narrow the country's large trade deficit, is still insignificant compared with agreements reached by Indonesia, Venezuela and Brazil, the trade sources said.

The trade deficit, which hit an estimated record 6.96 billion dlrs in 1985/86, is expected to decline to 5.6 billion in the current fiscal year.

But the push to include non-communist countries in countertrade is also due to other factors, including the slow growth of foreign reserves, a tight debt repayment schedule, shrinking aid and trade protectionism, businessmen said.

One source said India is showing more dynamism in promoting countertrade deals than in the past, when the deals were made discreetly because they break GATT rules. As a member of the General Agreement on Tariffs and Trade (GATT), India cannot officially support bartering.

The MMTC's recent countertrade deals include iron ore exports to Yugoslavia for steel structures and rails.

"MMTC's recent global tenders now include a clause that preference will be given to parties who accept payment in kind for goods and services sold to India," a trade official said, adding that the policy remains flexible.

"We also take into account other factors such as prices at which the goods and services are offered to India," the trade official said.

Early this year the commerce ministry quietly told foreign companies interested in selling aircraft, ships, drilling rigs and railway equipment to India that they stood a better chance if they bought Indian goods or services in return, the trade sources said.

Illustrating the point, the official said a South Korean firm recently agreed to sell a drilling platform worth 40 mln dlrs to the state-run Oil and Natural Gas Commission.

Noticia: 10406 Título: " INDIA STEPS UP COUNTERTRADE DEALS"

India is **searching** for **non-communist countertrade partners** to **help** it cut its trade **deficit** and **conserve** foreign exchange.

Wheat, **tobacco, tea, coffee, jute, engineering** and **electronic goods, as well as minerals** including **iron ore**, are all on offer in **return** for **crude** oil, **petroleum products, chemicals, steel** and **machinery**, trade sources told Reuters.

Most of the **impetus behind countertrade**, which **began** in 1984, **comes** from two state trading **firms** -- the State Trading Corp (**STC**) and the **Minerals** and **Metals** Trading Corp (**MMTC**).

"The two state trading **corporations** are **free** to **use** their **buying power** in **respect** to **bulk commodities** to **promote Indian exports**," a **commerce ministry spokeswoman** said, **adding** that **private firms** are **excluded** from **countertrading**.

One trade **source** said **India** has **targetted countries** that **depend** on an **Indian domestic market recently opened** to foreign **imports**.

However, **countertrade deals still** make up only a **small** part of **India**'s total trading and are **likely** to **account** for **less** than **eight** pct of the **estimated** 18.53 billion dlrs in trade
during the nine months **ended December**, the sources said.

Countertrade accounted for **just** five pct of **India**'s 25.65 billion dlrs in trade during **fiscal** 1985/86 **ended** March, against **almost nothing** in 1984/85, **official figures show**.

However, the **figures exclude exchanges** with the **Eastern Bloc paid** in **non-convertible Indian rupees**, the sources said.

Total trade with the **Soviet Union**, **involving swaps** of **agricultural produce** and **textiles** for **Soviet arms** and **crude** oil, is **estimated** at 3.04 billion dlrs in **fiscal** 1986/87, against three billion in 1985/86.

Anexo 3.

Ejemplo de algunas SFM's de la colección Reuters-21578

A continucación se muestran algunas SFM's extraídas con Dimasp-C_0 y β=15. El número entre paréntesis indica la frecuencia de la SFM.

SFM's de longitud 1

[15] BAHIA	[60] PORTS	[331]ANALYST	[203]SHIPMENTS
[358]REVIEW	[34] ROUTINE	[55] TYPES	[26] BRACKETS
[55] ZONE	[25] BUTTER	[63] EQUITIES	[16] BREAD
[68] DROUGHT	[43] DESTINATIONS	[229]BELIEVED	[17] SUNFLOWERSEED
[99] IMPROVING	[166]BUYERS	[519]LEAST	[108]DETAILED
[219]PROSPECTS	[113]ARGENTINA	[111]DEPRESSED	[57] HOTELS
[183]NORMAL	[47] MIDDAY	[136]MEAN	[31] MARATHON
[30] RESTORED	[70] MANAGE	[20] DILUTION	[44] ESTABLISHMENT
[209]WEEKLY	[96] OPERATED	[43] WILLIAMS	[160]FIELD
[69] BAGS	[33] BANCSHARES	[20] SHOCK	[65] IMPLICATIONS
[157]SEASON	[98] HOUSTON	[110]REACTION	[133]APPEARS
[121]STAGE	[158]APPLICATION	[134]EASE	[46] SEGMENT
[92] SEEMS	[224]EFFORT	[37] NEVERTHELESS	[96] APPROVES
[103]DELIVERED	[145]CREATE	[57] STANDS	[133]VOTED
[372]INCLUDED	[138]COUNTY	[81] LOSE	[63] RECOMMEND
[66] DOUBT	[91] LINK	[455]REDUCED	[68] TERMINAL
[21] HARVESTING	[201]HAVING	[64] JOSEPH	[120]INVOLVING
[60] STANDING	[43] BANKAMERICA	[311]POTENTIAL	[187]EQUAL
[26] THOUSAND	[120]DELAY	[41] FAILS	[130]EXCEED
[263]FARMERS	[95] RECOMMENDED	[45] RESTRUCTURED	[384]TECHNOLOGY
[57] PROCESSORS	[139]PORTION	[30] FARMER	[56] IMPROVEMENTS
[44] DOUBTS	[89] THREAT	[54] SORGHUM	[67] EXCLUSIVE
[17] EXPERIENCING	[35] NERVOUS	[125]REFLECTS	[203]WORLDWIDE
[46] OBTAINING	[22] WITHDRAWING	[61] COVERS	[198]MOVES
[34] SUPERIOR	[259]LONGER	[33] OILSEED	[303]OPERATION
[182]QUALITY	[28] ARTHUR	[84] GRAINS	[119]ENSURE
[55] RELUCTANT	[90] PROCEED	[43] OILSEEDS	[82] GENERATED
[187]SHIPMENT	[48] RECEIVES	[72] THOUSANDS	[94] FORMS
[28] BOOKED	[61] LAWRENCE	[97] SHOWING	[25] PRINTERS

SFM's de longitud 2

[84] NORTH AMERICA	[86] LOS ANGELES	[205]REAGAN ADMINISTRATION
[17] PUT OFF	[219]I THINK	[72] COMMERCE DEPARTMENT
[163]AS AS	[49] ELECTRIC POWER	[15] PHILLIPS PETROLEUM
[104]COMPANY EXPECTS	[42] SAN FRANCISCO	[18] CANADIAN COMPANY
[22] UNITED KINGDOM	[17] COMPUTER CHIP	[22] GOVERNMENT APPROVAL
[37] COMPANY DIVIDEND	[35] KANSAS CITY	[37] TEXAS AIR
[35] POLITICAL ANALYSTS	[27] COSTA RICA	[26] CONTINENTAL AIRLINES
[38] CHAIRMAN JOHN	[26] GULF WAR	[18] DEPARTMENT JUSTICE
[127]PRESS CONFERENCE	[23] MONEY CENTER	[23] FEDERAL REGULATORS
[18] DEFENSE SECRETARY	[22] VERY WELL	[26] GULF MEXICO
[16] DONALD REGAN	[184]HONG KONG	[54] INSURANCE COMPANIES

[17] COMMERCE MINISTRY
[28] COFFEE EXPORT
[20] COFFEE MARKET
[16] ITALIAN TREASURY
[30] PRECIOUS METALS
[97] SAUDI ARABIA
[18] TEXACO TX
[25] PETROLEUM PRODUCTS
[22] CRUDE EXPORTS
[39] COMPUTER SYSTEMS
[19] WELL KNOWN
[30] VERY DIFFICULT
[32] I BELIEVE
[44] PERSONAL COMPUTER
[21] VERY SERIOUS
[76] SOUTH AFRICAN
[26] GAS EXPLORATION
[18] AMERICAN AIRLINES
[18] HOME SHOPPING
[37] CABLE TELEVISION
[23] DATA PROCESSING
[89] LATIN AMERICAN
[20] INTERNATIONAL AIRPORT
[39] VERY GOOD
[28] AIRBUS INDUSTRIE
[16] PRESIDENT JACQUES
[21] VERY IMPORTANT
[33] VOLKSWAGEN AG
[25] TOYOTA MOTOR
[26] VERY MUCH
[16] BRITISH AIRWAYS
[29] COCA COLA

SFM's de longitud 3

[24] AMERICAN EXPRESS AXP
[20] PRESIDENT REAGAN VETO
[15] I AM SURE
[20] INTERNATIONAL COFFEE ORGANISATION
[20] WALL STREET ANALYSTS
[15] CONSERVATION RESERVE PROGRAM
[15] FEDERAL RESERVE INTERVENE
[19] NATURAL GAS LIQUIDS
[16] NATURAL GAS RESERVES
[18] USDA AGRICULTURE DEPARTMENT
[15] CANADIAN IMPERIAL COMMERCE
[32] NATIONAL STATISTICS INSTITUTE
[15] NUCLEAR REGULATORY COMMISSION
[19] AS MANY AS
[15] DAYS DAYS DAYS
[16] WEST GERMAN GOVERNMENT
[70] I DON THINK
[25] I DON KNOW
[19] I DON SEE
[15] AS FAST AS
[15] AMERICAN MOTORS AMO
[38] INTERNATIONAL MACHINES IBM
[24] NUCLEAR POWER PLANT
[22] CHICAGO MERCANTILE CME
[38] AS FAR AS
[38] AS LOW AS
[18] JAPANESE FINANCIAL INSTITUTIONS
[18] AMERICAN PETROLEUM INSTITUTE
[17] TONNES WHITE SUGAR
[33] RIO DE JANEIRO
[18] PRESIDENT RONALD REAGAN
[36] PRESIDENT JOSE SARNEY
[20] PETROLEOS DE VENEZUELA

SFM's de longitud 4

[20] EXPORT ENHANCEMENT PROGRAM INITIATIVE
[20] EXPORT ENHANCEMENT PROGRAM EEP
[25] WHITE HOUSE CHIEF STAFF
[24] WHITE HOUSE MARLIN FITZWATER
[19] CHIEF STAFF HOWARD BAKER
[33] INTERNATIONAL COFFEE ORGANIZATION ICO
[20] EXECUTIVE VICE PRESIDENT CHIEF
[16] TREASURY BALANCES FEDERAL RESERVE
[20] FINANCE MINISTER MICHAEL WILSON
[40] FINANCE MINISTER EDOUARD BALLADUR
[39] FEDERAL SAVINGS LOAN INSURANCE
[26] CHIEF EXECUTIVE OFFICER COMPANY
[17] CUBIC FEET NATURAL GAS
[49] INTERNATIONAL MONETARY FUND IMF
[27] GROSS DOMESTIC PRODUCT GDP
[44] AGRICULTURE SECRETARY RICHARD LYNG

[28] PRIME MINISTER JACQUES CHIRAC
[22] FINANCE SECRETARY JAIME ONGPIN
[17] UNITED ARAB EMIRATES UAE
[19] BOUGHT TONNES CANADIAN RAPESEED
[31] EUROPEAN CURRENCY UNITS ECUS
[40] AS SOON AS POSSIBLE
[19] BANQUE NATIONALE DE PARIS
[17] CUSTOMER REPURCHASE AGREEMENTS FED
[20] CUSTOMER REPURCHASES FEDERAL RESERVE
[33] COMMODITY FUTURES COMMISSION CFTC
[20] FEDERAL ENERGY REGULATORY COMMISSION
[16] BRAZILIAN COFFEE INSTITUTE IBC
[17] NASDAQ NATIONAL MARKET SYSTEM
[37] MINISTRY INTERNATIONAL INDUSTRY MITI
[15] EUROPEAN MONETARY SYSTEM EMS
[16] UNITED FOOD COMMERCIAL WORKERS
[15] AS QUICKLY AS POSSIBLE
[30] INTERNATIONAL COCOA ORGANIZATION ICCO
[17] ENERGY SECRETARY JOHN HERRINGTON

SFM's de longitud 5

[30] AS PRESIDENT CHIEF EXECUTIVE OFFICER
[71] FEDERAL RESERVE CHAIRMAN PAUL VOLCKER
[28] AS CHAIRMAN CHIEF EXECUTIVE OFFICER
[15] CONVERTIBLE SUBORDINATED DEBENTURES PROCEEDS USED
[24] BRAZILIAN FINANCE MINISTER DILSON FUNARO
[27] ORGANISATION ECONOMIC COOPERATION DEVELOPMENT OECD
[34] RULING LIBERAL DEMOCRATIC PARTY LDP
[21] POINTS LONDON INTERBANK OFFERED LIBOR
[35] NON CALLABLE BOND AVAILABLE DENOMINATIONS
[17] NON CALLABLE RATED MOODY INVESTORS
[18] BOND AVAILABLE DENOMINATIONS LISTED LUXEMBOURG
[19] BILLS MATURING OFFICIAL HANDS TREASURY
[18] JAPAN WEST GERMANY FRANCE BRITAIN
[15] ASSISTANCE ENGLAND PROVIDED MONEY MARKET
[15] CORPORATE AFFAIRS MINISTER MICHAEL HOWARD
[15] ISSUING AUSTRALIAN EUROBOND PAYING PRICED
[19] HELP ENGLAND PROVIDED MONEY MARKET
[18] BUNDESBANK PRESIDENT KARL OTTO POEHL
[19] JAPANESE PRIME MINISTER YASUHIRO NAKASONE
[17] BRITISH PRIME MINISTER MARGARET THATCHER
[34] JAPANESE FINANCE MINISTER KIICHI MIYAZAWA

SFM's de longitud 6

[15] UNITED STATES JAPAN WEST GERMANY FRANCE
[15] SUBORDINATED DEBENTURES COUPON PAR PRICING LEAD
[16] WEST GERMAN FINANCE MINISTER GERHARD STOLTENBERG
[15] EUROBOND PAYING INDICATED COUPON PRICED PAR

```
[15] BOND AVAILABLE DENOMINATIONS YEN LISTED LUXEMBOURG
[15] AMONG MAIN FACTORS AFFECTING LIQUIDITY BILLS
[15] ENGLAND PROVIDED MONEY MARKET STG ASSISTANCE
[26] FEES COMPRISE SELLING CONCESSION UNDERWRITING COMBINED
[16] BRITAIN CANADA FRANCE JAPAN WEST GERMANY
```

SFM's de longitud 7

```
[19] FILED COMMISSION REGISTRATION COVERING CONVERTIBLE SUBORDINATED
     DEBENTURES
[16] PREVIOUS TREASURY LATEST BUDGET BALANCES LOAN ACCOUNTS
[16] RESPECTIVE DAYS TREASURY OPERATING BALANCE TOTALED COMPARED
[24] OFFERING CONVERTIBLE SUBORDINATED DEBENTURES COUPON PAR PRICING
[15] ISSUING YEN EUROBOND PAYING PRICED LEAD MANAGER
[15] LISTED LUXEMBOURG FEES COMPRISE SELLING CONCESSION UNDERWRITING
[19] REPURCHASES FEDERAL RESERVE ENTERED GOVERNMENT MARKET ARRANGE
[15] COMMISSION REGISTRATION COVERING CONVERTIBLE SUBORDINATED DEBENTURES
     PROCEEDS
[15] MIDWEST GRAIN FUTURES EDT MINNEAPOLIS WHEAT JUL
```

SFM's de longitud 8

```
[16] NOTES COUPON PRICED YIELD BASIS POINTS COMPARABLE TREASURY
[22] DEBENTURES CONVERTIBLE COMPANY REPRESENTING PREMIUM NON CALLABLE RATED
```

SFM's de longitud 9

```
[15] FEDERAL RESERVE ENTERED GOVERNMENT MARKET ARRANGE CUSTOMER REPURCHASE
     AGREEMENTS
[18] COUPON PRICED YIELD BASIS POINTS COMPARABLE TREASURY NON CALLABLE
[23] LISTED LUXEMBOURG SELLING CONCESSION UNDERWRITING COMBINED PAYS PAYMENT
     DATE
[15] REPURCHASE AGREEMENTS FED DEALERS FEDERAL FUNDS FED BEGAN TEMPORARY
```

SFM's de longitud 10

```
[16] SHORT DISCOUNT NOTES AS FOLLOWS MATURITY OLD MATURITY DAYS DAYS
```

SFM's de longitud 11

```
[25] PROPOSED OFFERINGS RECENTLY FILED SEC FOLLOWING PROPOSED OFFERINGS
     FILED RECENTLY COMMISSION
```

SFM's de longitud 25

```
[17] HOG CATTLE SLAUGHTER GUESSTIMATES CHICAGO MERCANTILE FLOOR TRADERS
     COMMISSION HOUSE REPRESENTATIVES GUESSTIMATING HOG SLAUGHTER HEAD
     VERSUS AGO AGO CATTLE SLAUGHTER GUESSTIMATED HEAD VERSUS AGO AGO
```

Anexo 4.

Ejemplo de algunas SFM's en un solo documento

Utilizando el documento *Autobiography* de la colección Alex, se buscaron SFM's con β=2 con Dimasp-D_0. Los siguientes extractos pertenecen a las dos ocurrencias en el documento original donde se presentó una SFM de longitud 35. En este ejemplo es posible ver que los dos extractos tienen 2 oraciones. Cabe señalar que, si las oraciones del documento *Autobiography* fueran consideradas como una colección de documentos entonces Dimasp-C_n no podría encontrar esta SFM.

Extracto 1:

In the impeachment of judge Pickering of New Hampshire, a habitual & maniac drunkard, no defence was made. Had there been, the party vote of more than one third of the Senate would have acquitted him.

Extracto 2:

(* 10) **In the impeachment of judge Pickering of New Hampsire, a habitual & maniac drunkard, no defence was made. Had there been, the party vote of more than one third of the Senate would have acquitted him**.

Utilizando el documento *Letters* de la colección Alex, se buscaron SFM's con β=2 con Dimasp-D_0. Los siguientes extractos pertenecen a las dos ocurrencias en el documento original donde se presentó una SFM de longitud 21. En este ejemplo es posible ver que en el primer extracto la SFM se presentó dentro de una oracion. Sin embargo, en el segundo extracto la SFM se presentó en 5 oraciones. Cabe señalar que, si las oraciones del documento *Letters* fueran consideradas como una colección de documentos entonces Dimasp-C_n no podría encontrar esta SFM.

Extracto 1:

Thus Finch quotes Prisot; Wingate also; **Sheppard quotes Prisot, Finch and Wingate; Hale cites nobody; the court in Woolston's case cite Hale; Wood cites Woolston's case; Blackstone** that and Hale; and Lord Mansfield, like Hale, ventures it on his own authority.

Extracto 2:

Sheppard quotes Prisot,Finch and Wingate. Hale cites nobody. The court in Woolston's case, cite Hale. Wood cites Woolston's case. Blackstone quotes Woolston's case and Hale. And Lord Mansfield, like Hale, ventures it on his own authority.

Utilizando el documento *Letters* de la colección Alex, se buscaron SFM's con β=2 con Dimasp-D_0. Los siguientes extractos pertenecen a las dos ocurrencias en el documento original donde se presentó una SFM de longitud 354. En ambos extractos la SFM se presenta en 10 oraciones. Cabe señalar que, si las oraciones del documento *Letters* fueran consideradas como una colección de documentos entonces Dimasp-C_n no podría encontrar esta SFM.

Extracto 1:

Ever since that I have been paying twelve hundred dollars a year interest on his debt, **which, with my own, was absorbing so much of my annual income, as that the maintenance of my family was making deep and rapid inroads on my capital, and had already done it. Still, sales at a fair price would leave me competently provided. Had crops and prices for several years been such as to maintain a steady competition of substantial bidders at market, all would have been safe. But the long succession of years of stunted crops, of reduced prices, the general prostration of the farming business, under levies for the support of manufactures, &c., with the calamitous fluctuations of value in our paper medium, have kept agriculture in a state of abject depression, which has peopled the western States by silently breaking up those on the Atlantic, and glutted the land market, while it drew off its bidders. In such a state of things, property has lost its character of being a resource for debts. Highland in Bedford, which, in the days of our plethory, sold readily for from fifty to one hundred dollars the acre, (and such sales were many then,) would not now sell for more than from ten to twenty dollars, or one-quarter or one-fifth of its former price. Reflecting on these things, the practice occurred to me, of selling, on fair valuation, and by way of lottery, often resorted to before the Revolution to effect large sales, and still in constant usage in every State for individual as well as corporation purposes. If it is permitted in my case, my lands here alone, with the mills, &c., will pay every thing, and leave me Monticello and a farm free. If refused, I must sell everything here, perhaps considerably in Bedford, move thither with my family, where I have not even a log hut to put my head into, and whether ground for burial, will depend on the depredations which, under the form of sales, shall have been committed on my property. The question then with me was _ultrum horum_? But why afflict you with these details? Indeed, I cannot tell, unless pains are lessened by communication with a** frt, which, with my own, was absorbing so much of my annual income, as that the maintenance of my family was making deep and rapid inroads on my capital, and had already done it.

Extracto 2:

The question then with me was _ultrum horum_? But why afflict you with these details? Indeed, I cannot tell, unless pains are lessened by communication with a frt, **which, with my own, was absorbing so much of my annual income, as that the maintenance of my family was making deep and rapid inroads on my capital, and had already done it. Still, sales at a fair price would leave me competently provided. Had crops and prices for several years been such as to maintain a steady competition of substantial bidders at market, all would have been safe. But the long succession of years of stunted crops, of reduced prices, the general prostration of the farming business, under levies for the support of manufactures, &c., with the calamitous fluctuations of value in our paper medium, have kept agriculture in a state of abject depression, which has peopled the western States by silently breaking up those on the Atlantic, and glutted the land market, while it drew off its bidders. In such a state of things, property has lost its character of being a resource for debts. Highland in Bedford, which, in the days of our plethory, sold readily for from fifty to one hundred dollars the acre, (and such sales were many then,) would not now sell for more than from ten to twenty dollars, or one-quarter or one-fifth of its former price. Reflecting on these things, the practice occurred to me, of selling, on fair valuation, and by way of lottery, often resorted to before the Revolution to effect large sales, and still in constant usage in every State for individual as well as corporation purposes. If it is permitted in my case, my lands here alone, with the mills, &c., will pay every thing, and leave me Monticello and a farm free. If refused, I must sell everything here, perhaps considerably in Bedford, move thither with my family, where I have not even a log hut to put my head into, and whether ground for burial, will depend on the depredations which, under the form of sales, shall have been committed on my property. The question then with me was _ultrum horum_? But why afflict you with these details? Indeed, I cannot tell, unless pains are lessened by communication with a** friend.