Diseño e Implementación de un Sistema de Búsqueda en una Colección de Texto Comprimido

Carlos Avendaño Pérez, Claudia Feregrino Uribe, Gonzalo Navarro Badino.
Coordinación de Ciencias Computacionales
carlosap@ccc.inaoep.mx, cferegrino@inaoep.mx, gnavarro@dcc.uchile.cl

RESUMEN

Este trabajo se enfoca al problema de búsqueda de patrones en texto comprimido. La meta es buscar un patrón en el texto sin descomprimirlo. Este es un problema sumamente relevante, ya que nos permite mantener bases de datos de texto comprimidas y realizar búsquedas eficientes en ellas. Se ha mostrado en la literatura que los algoritmos de compresión presentados en este trabajo, logran buenas razones de compresión y son rápidos para comprimir y descomprimir los datos. Los algoritmos para búsqueda de patrones permiten búsquedas exactas y aproximadas directamente sobre texto comprimido. El objetivo final es obtener un software de búsqueda directa sobre textos comprimidos con los comandos *compress* y *gzip*, los cuales emplean los algoritmos de compresión LZW y LZ77 respectivamente.

I INTRODUCCIÓN

El considerable aumento de información textual disponible en diversos medios tales como librerías digitales, bases de datos de documentos y la web, conlleva el problema de almacenar esta información y/o transmitirla sobre un medio de comunicación eficientemente. Aunque la capacidad de los dispositivos actuales para almacenar información crece rápidamente, el tamaño de las colecciones lo hacen en una medida más grande. La compresión de texto es una opción para reducir el espacio necesario para el almacenamiento y el tiempo para transmitir los datos. Al mismo tiempo, la compresión de texto es un tema importante en el contexto de los sistemas de búsqueda de patrones, dado que toma menos tiempo buscar directamente sobre texto comprimido que sobre texto sin comprimir. El precio a pagar es el tiempo necesario para codificar y decodificar el texto.

A pesar de las ventajas ofrecidas por los métodos de compresión, la mayoría de las aplicaciones que manipulan bases de datos de texto no procesan textos comprimidos, esto se debe a que los algoritmos de compresión de texto son incompatibles con los algoritmos convencionales de búsqueda utilizados en los sistemas de búsqueda de texto. De resultados mostrados en la literatura se sabe que el procesamiento de texto comprimido puede mejorar el desempeño de los sistemas de búsqueda.

De forma general, la compresión de texto consiste en eliminar la redundancia del texto para representarlo con un número menor de bits. En la práctica los algoritmos más utilizados para comprimir texto son los de la familia Ziv-Lempel debido a que se logran buenas razones de compresión con velocidades de compresión/descompresión eficientes.

Por otra parte, la búsqueda de patrones dentro de un texto es un problema muy recurrente en el ámbito del estudio de las ciencias computacionales. Este problema se define formalmente de la siguiente manera:

> Dado un patrón $P = p_1 \ldots p_m$ y un texto $T = t_1 \ldots t_u$, ambas secuencias de caracteres sobre el alfabeto $\sum$, encontrar todas las ocurrencias de P en T.

La combinación de compresión de texto y búsqueda de texto resulta interesante para algunas aplicaciones, tales como las bases de datos textuales, debido a que en éstas los textos deben mantenerse comprimidos para reducir el espacio necesario para su almacenamiento, y además se necesitan búsquedas eficientes. Este problema se conoce como búsqueda directa en texto comprimido y fue definido por primera vez en el trabajo de Amir y Beson [1] de la siguiente manera:

> Dado un texto $T = t_1 \ldots t_u$, cuyo texto comprimido correspondiente es $Z = z_1 \ldots z_n$, y un patrón $P = p_1 \ldots p_m$, encontrar todas las ocurrencias de P y T, usando solamente P y Z.

Continuando con esta línea de investigación, se propone utilizar las dos técnicas mencionadas anteriormente con el objetivo de obtener un

software de búsqueda directa sobre textos comprimidos con *compress* y *gzip* que se comporte lo más parecido a *grep*, de manera que se reemplace a *zgrep* y que sea flexible en la especificación del patrón, permitiendo tanto búsqueda exacta como aproximada de texto.

II CONCEPTOS BÁSICOS

A continuación, se da una introducción sobre los conceptos básicos para comprender mejor este trabajo.

En búsqueda de texto los dos algoritmos secuenciales más conocidos para solucionar el problema de búsqueda exacta de patrones es el de Knuth-Morris-Pratt (KMP) [7] y el de Boyer-Moore(BM) [4]. La idea de estos algoritmos consiste en utilizar heurísticas para reducir el número de comparaciones entre caracteres realizadas durante la búsqueda. Existe otro trabajo, que es un nuevo paradigma para resolver dicho problema, utilizando procesamiento de paralelismo de bits propuesto por Baeza-Yates y Gonnet [3]. La ventaja de esta técnica es que el procesador desempeña algunas operaciones en paralelo sobre todos los bits de una palabra de computadora, lo cual acelera significativamente el proceso de búsqueda.

Otro concepto es la compresión de texto, que consiste de dos etapas distintas: *modelado*, donde se determina una manera lógica de dividir el texto en un conjunto de símbolos, y se estima una probabilidad de cada símbolo modelado a ocurrir en el texto, y la *codificación*, que define cómo representar en el texto comprimido cada símbolo modelado. Los métodos de compresión pueden ser clasificados de acuerdo al modelo de datos utilizado, pudiendo ser estáticos, semi-estáticos y adaptables. Los métodos estáticos son métodos que utilizan el mismo modelo para todos los textos procesados. Esta forma de modelado se utiliza solamente cuando la velocidad y simplicidad son los únicos requisitos exigidos. Un método semi-estático utiliza un modelo diferente para cada texto. Antes de comprimir, analiza el texto y construye un modelo que será utilizado tanto por el codificador como por el decodificador. Un modelo adaptable, actualiza un modelo (a partir de un modelo inicial) a cada lectura de un símbolo, evitando así la necesidad de un análisis previo de todo el texto. De estos tipos de modelado, el más conveniente para sistemas de búsqueda de texto es el semi-estático, por que permite comenzar a buscar en el texto comprimido desde cualquier parte del texto. Sin embargo, las colecciones de texto generalmente son comprimidas por métodos de la familia Ziv-Lempel (los cuales utilizan un modelo adaptable), es por eso que es necesario el estudio de técnicas de búsqueda en texto comprimido con estos métodos.

Otra forma de clasificar los métodos de compresión es dividirlos de acuerdo con el método de codificación utilizado. En este caso, se dividen los métodos en estadísticos y de diccionario. Los primeros asignan un código a cada símbolo del texto, basados en la probabilidad de ocurrencia de cada símbolo. Cuanto mayor es la probabilidad de ocurrencia del símbolo, menor es la longitud del código que lo representa. Los métodos estadísticos más populares son la codificación de Huffman y la codificación aritmética. Los siguientes métodos, de diccionario, son denominados así por crear un diccionario de datos durante el proceso de codificación. La codificación se realiza sustituyendo grupos de caracteres consecutivos (frases) por apuntadores a las entradas del diccionario. Lo compresión se logra al sustituir una frase por un apuntador que necesita menos espacio. No hay diferencia entre modelado y codificación en los métodos de diccionario y no hay probabilidades explicitas asociadas a las frases. Los métodos de diccionario más populares son los de la familia Ziv-Lempel.

Como se mencionó anteriormente, uno de los métodos de compresión más utilizado por los sistemas de búsqueda de texto es la codificación de Huffman [5]. La idea de este método es comprimir el texto asignando códigos más pequeños a símbolos con altas frecuencias de ocurrencias. El proceso de compresión se logra en dos pasos: 1) El codificador hace un recorrido sobre el texto para obtener la frecuencia de cada palabra diferente en el texto. 2) Efectúa la compresión en base a dichas frecuencias. Dado que es un método de compresión semi-estático, se puede comenzar a buscar en el texto comprimido desde cualquier parte del texto.

Otro método importante para compresión de texto es el de la familia Ziv-Lempel. LZ77 [12], LZ78 [13] y LZW [11] son las variantes más utilizadas actualmente en compresión de datos genéricos y también presentan buenos resultados cuando son aplicadas a compresión de textos. Los métodos de Ziv y Lempel son métodos de diccionario. La principal característica de estos métodos es que son bastante simples y eficientes, tanto en tiempo como en razones de compresión. El problema de estos métodos es que no se pueden realizar búsquedas aleatorias, teniendo que empezar la

búsqueda desde el principio del texto. Sin embargo es posible intentar un desplazamiento con los caracteres explícitamente codificados y descartar la mayor parte del texto, acelerando el proceso de búsqueda [2,3].

III TRABAJO RELACIONADO

Normalmente, las bases de datos textuales se almacenan en forma de archivos comprimidos para reducir el espacio necesario para el almacenamiento y el tiempo para transmitir los datos. Por consiguiente, surge la necesidad de implementar técnicas eficientes para la búsqueda sobre texto comprimido. La forma tradicional de satisfacer esta necesidad consiste en descomprimir primero y después utilizar un algoritmo tradicional de búsqueda de texto. Sin embargo, el hecho de descomprimir previamente incrementa un costo de entrada/salida. De ahí nace el problema conocido como "búsqueda directa sobre texto comprimido".

En el trabajo de Amir y Beson [1], se presenta un método de búsqueda para archivos comprimidos con LZ77. Desde entonces varios trabajos se han desarrollado para dar distintas soluciones a este problema. Cuando la colección de documentos se comprime con un método de compresión semi-estático, se distinguen dos problemas. El primero es que se requiere un gran espacio de memoria en tiempo de codificación para almacenar todas las palabras distintas que están contenidas en la colección. El segundo problema es que un modelo semi-estático asume que se conoce el texto completo por adelantado. Sin embargo, nuevos documentos pueden ser agregados a la colección. En este caso el esquema de compresión debe permitir al texto ser dinámico, y no tener que recomprimir la colección después de cada inserción. En [8] se dan algunas soluciones a estos dos problemas. En [2] fue presentado un algoritmo que busca en archivos comprimidos con LZW. Otros trabajos recientes presentan nuevos métodos para búsqueda en textos comprimidos por algoritmos derivados de LZ77 y LZ78. Uno de estos trabajos es el de Kida *et al*. [6] donde presentan un algoritmo práctico para búsqueda en archivos comprimidos con LZ77. Otro es el de Navarro y Raffinot [9] donde presentan un algoritmo genérico para búsqueda en archivos comprimidos con LZ77, LZ78 y sus derivados.

IV SOLUCIÓN PROPUESTA

En este trabajo se propone diseñar e implementar un sistema de búsqueda de patrones en una colección de documentos de texto comprimido, que tenga la capacidad de hacer las búsquedas directamente sobre el texto comprimido sin necesidad de descomprimir la colección, puesto que toma menos tiempo buscar directamente sobre texto comprimido que sobre texto sin comprimir.

Las formulaciones de las consultas (basadas sobre el concepto de patrón) permitirán recuperar partes del texto que tienen alguna propiedad. Un patrón es un conjunto de características sintácticas que deben ocurrir en un segmento de texto. Aquellos segmentos que satisfacen las especificaciones del patrón se les denominan "*igualdad*" del patrón. El interés está en los documentos que contengan segmentos que *igualen* la búsqueda de un patrón dado. La meta es implementar un sistema de búsqueda que sea flexible en cuanto a la especificación de algunos tipos de patrones, que vayan desde un rango muy simple (por ejemplo palabras), a más complejas (tales como las expresiones regulares). Sin embargo, entre más poderoso sea el conjunto de patrones permitido, más complicadas son las consultas que el usuario puede formular y más compleja es la implementación de la búsqueda.

En el trabajo [10], se desarrolló un software que tiene una interfaz muy parecida a *grep* y que permite búsqueda de patrones tales como: *Palabras* (secuencia de caracteres), *prefijos (*una cadena que forma el inicio de una palabra en el texto), *sufijos* (una cadena que forma la terminación de una palabra en el texto) y *subcadenas* (una cadena la cual puede aparecer dentro de una palabra en el texto). La idea es continuar con ese trabajo y extenderlo para que soporte búsqueda aproximada de patrones simples y búsqueda de expresiones regulares. A continuación explicaremos estos dos tipos de búsqueda con mayor detalle.

Búsqueda aproximada: Una palabra con un margen de error. Esta búsqueda de patrones recupera todas las palabras en el texto que sean "similares" a una palabra dada. El concepto de similaridad puede ser definido de varias maneras. El concepto general es que el patrón o el texto pueden tener errores (ya sea al mecanografiar, a partir software de reconocimiento óptico de caracteres, u otros) y la consulta deberá tratar de recuperar la palabra dada y sus variantes erróneas. El modelo mejor aceptado para la similaridad entre palabras en la búsqueda de texto es la distancia de edición. La distancia de edición entre dos palabras se define como el menor número de inserciones, eliminaciones o reemplazos de caracteres necesarios para hacer iguales a las dos

palabras. Este modelo también puede ser extendido a búsqueda de subcadenas (no solo palabras), recuperando cualquier segmento de texto que esté en la distancia de edición permitido para la búsqueda de un patrón.

Expresiones regulares: Algunos sistemas de búsqueda de texto permiten búsqueda para expresiones regulares. Una expresión regular es un patrón más general construido por simples cadenas (lo cual significa que serán igualados como subcadenas) y los siguientes operadores: *Unión:* si e_1 y e_2 son expresiones regulares, entonces (e_1| e_2) igualan a los que e_1 o e_2 igualan. *Concatenación:* si e_1 y e_2 son expresiones regulares, las ocurrencias de (e_1 e_2) están formadas por las ocurrencias de e_1 inmediatamente seguidos por e_2. *Repetición:* si *e* es una expresión regular, entonces (*e**) iguala a una secuencia de cero o más ocurrencias contiguas de *e*.

Este trabajo se encuentra en sus primeras etapas, en donde hasta el momento se han estudiado diversos algoritmos de compresión de texto y algoritmos de búsqueda aproximada y búsqueda de expresiones regulares en texto. El objetivo que se desea lograr al finalizar este trabajo es obtener un software de búsqueda directa sobre textos comprimidos que se comporte lo más parecido a *grep* (comando de UNIX para búsqueda de patrones en texto sin comprimir) de manera de reemplazar a *zgrep* (comando que busca patrones sobre texto comprimido, descomprime el texto y utiliza grep para efectuar la búsqueda haciendo que esta sea más lenta), y que sea flexible en la especificación de la búsqueda, permitiendo tanto búsqueda exacta como aproximada.

V CONCLUSIONES

La cantidad de información textual accesible en diversos medios se irá incrementando día con día, de ahí que las colecciones de texto se mantengan comprimidas. Por consiguiente es muy importante la existencia de un sistema de búsqueda que nos permita acceder a cierta información relevante mediante consultas flexibles sin necesidad de descomprimir la colección. Sin embargo, en la actualidad no existe un sistema que nos ofrezca tal flexibilidad, haciendo problemático encontrar la información deseada o inclusive haciendo que no se pueda recuperar información relevante.

VI AGRADECIMIENTOS

Se agradece al CONACyT el apoyo otorgado a través de la Beca para Estudios de Maestría #177938.

VII REFERENCIAS

[1] Amir A. and Benson G. “Efficient two-dimensional compressed matching”. In Proc. Second IEEE Data Compression Conference, pages 279-288, 1992.

[2] Amir A. and Benson G. and Farach M. “Let sleeping files lie: pattern matching in z-compressed file”. Journal of Computer and Systems Sciences, 52(2):299-307, 1996.

[3] Baeza-Yates R. and Gonnet G. “A new approach to text searching”. Communications of the ACM, 35(10):74-82, 1992.

[4] Boyer R.S. and Moore J.S. “A fast string searching algorithm”. Communications of the ACM, 20(10):762-772, 1977.

[5] Huffman D. A. “A method for the construction of minimum-redundancy codes”. In Proc. of the Institute of Electrical and Radio Engineers, volume 40, pages 1090-1101, 1952.

[6] Kida T., Takeda M., Shinohara A., Miyazaki M. and Arikawa S. “Multiple pattern matching in lzw compressed text”. In Proc. Eighth IEEE data Compression Conference, 1998.

[7] Knuth D., Morris J. and Pratt V. “Fast pattern matching in strings”. Siam Journal on Computing”, 6(2):323-350, 1977.

[8] Moffat A., Zobel J. and Sharman N. “Text Compression for Dynamic Document Databases”. IEEE Transactions on Knowledge and Data Engineering. 1(2), 1997.

[9] Navarro G. and Raffinot M. “A general practical approach to pattern matching over Ziv-Lempel compressed text”. In Proc. of the 10th symp. On Combinatorial Pattern Matching, Lecture Notes in Computer Science, Springer-Verlag, 1999.

[10] Rojas M. E. “Búsqueda tipo Boyer-Moore sobre texto comprimido con LZW”. Tesis de licenciatura, departamento de Ciencias de la Computación, Universidad de Chile, 2003.

[11] Welch T. A. “A technique for high-performance data compression”. IEEE Computer, 17(6):8-19, 1984.

[12] Ziv. J and Lempel A. “A universal algorithm for sequential data compression”. IEEE Transactions on Information Theory, 23(3):337-343, 1977.

[13] Ziv. J and Lempel A. “Compression of individual sequences via variable-rate coding”. IEEE Transactions on Information Theory, 24(5):530-536, 1978.