PROYECTO FINAL CP 2014-I (Lab2)

BÚSQUEDA DE PATRONES - IMPLEMENTACIÓN CÓDIGO DE LA BIBLIA

TABLA DE CONTENIDOS

1. ANTECEDENTES GENERALES

1.1. INTRODUCCIÓN

El código de la Biblia, también conocido como el código de la Torá, consiste en un supuesto código oculto en la Torá judía *(el Pentateuco de la Biblia cristiana)* que relata acontecimientos del pasado, presente y futuro. Estos códigos son legibles gracias a unas reglas de codificación que únicamente pueden aplicarse al texto en hebreo antiguo utilizando programas informáticos. El libro, El código secreto de la Biblia de Michael Drosnin, publicado en 1997, popularizó ésta teoría.
Algunos de los más expertos decodificadores han hecho sus propias pruebas y han encontrado la veracidad de mensajes ocultos dentro de los textos en la Biblia frente a que otros han negado dicha posibilidad, entre los que se incluye uno de los mayores estudiosos del código: Robert J. Aumann *(Premio Nobel de Economía en 2005).*

1.2. El CÓDIGO

El principal método por el cual se extraen los mensajes significativos es la secuencia de letras equidistantes *(ELS, para una distancia "d" y un tamaño de matriz regular **n**x**m**, variables).* Para obtener una ELS de un texto, se escoge un punto de partida (cualquier letra) y una distancia *(un número, preferentemente negativo).* Entonces, comenzando en el punto de partida, se seleccionan letras del texto equidistantes, separadas por el número que se haya elegido para la distancia. Por ejemplo, en la oración: **"entre cojos osados",** las letras en negrita leídas de izquierda a derecha, separadas por una distancia de "**d=4**" letras, forman la palabra ECOS. Los espacios y los signos de puntuación se deben ignorar.

A menudo varias secuencias relacionadas con el mismo tema pueden aparecer simultáneamente en una serie de letras. Esto se debe a que se coloca el texto en una matriz regular, con el mismo número de letras en cada línea, extrayendo después un rectángulo. En el ejemplo, se muestra parte del Génesis (26, 5–10) con 33 letras por línea. Se muestran secuencias para BIBLIA Y CÓDIGO *("BIBLE", ""CODE").* Normalmente sólo haría falta mostrar un rectángulo más pequeño, como el que aparece dibujado en la figura. En ese caso habría letras que faltarían, pero es esencial que el número de letras que falten sea el mismo para cada par de líneas adyacentes. Si no, no se cumpliría la secuencia.

Aunque se haya mostrado un ejemplo en inglés, para hacer una búsqueda de forma correcta habría que usar el texto bíblico en hebreo. Por motivos religiosos, la mayor parte de los defensores judíos del código usan sólo la Torá *(los cinco primeros libros de La Biblia).* Además, ya que las traducciones a cualquier otro idioma *(de las cuales hay cientos de versiones para escoger)* no son el texto original de la

Biblia, esto requeriría que se creyera en el creacionismo de los idiomas (*por la influencia de una entidad omnisciente, o gracias a una cuidadosa construcción)* de modo que secuencias tan complejas como las encontradas en la Torá hebrea estuvieran presentes también en cada traducción. Otra alternativa consistiría en admitir que las secuencias halladas por los estudiosos del código no sean tan complejas o tan difíciles de encontrar como se dice.

1.3. EJEMPLOS

1.3.1. Ejemplo Nº1:

Para buscar las secuencias de letras equidistantes en un texto, se eliminan los espacios y marcas de puntuación sin alterarse el orden de las palabras, de manera tal que el texto se convierte en una hilera continua de letras. Por ejemplo, considerese las primeras 200 letras del clásico literario "Moby Dick" en inglés:

CALL ME ISHMAEL. SOME YEARS AGO--NEVER MIND HOW LONG PRECISELY--
HAVING LITTLE OR NO MONEY IN MY PURSE, AND NOTHING PARTICULAR TO
INTEREST ME ON SHORE, I THOUGHT I WOULD SAIL ABOUT A LITTLE AND SEE
THE WATERY PART OF THE WORLD. IT IS A WAY I HAVE OF DRIVING

El texto queda de la siguiente manera:

CALLMEISHMAELSOMEYEARSAGONEVERMINDHOWLONGPRECISELYHAVINGLITTLE..”

Ahora, se está en condiciones para buscar la palabra deseada. Por ejemplo, se busca la palabra "sol" y el algoritmo debe informar que sólo aparece unas seis veces; es decir que la palabra "sol" sólo es posible encontrarla en forma de secuencia de letras equidistantes solamente unas seis veces en todo el texto.
Para el presnete caso, se ha escogido sólo uno de esos seis resultados. La palabra "sol" puede ser encontrada leyendo de atrás hacia adelante en todo el texto con un salto de 24 letras:

“..INGLITT**L**EORNOMONEYINMYPURSEANDN**O**THINGPARTICULARTOINTERE**S**T..”

El algoritmo se implementa considerando la estructura de datos “Matriz” y para una “ventana” ejemplo de “9x24”, la matriz presenta la siguiente estructura, en la cual el término “SOL”, comprende, para este caso, las filas N°5, 4 y 3 y la columna N°13.

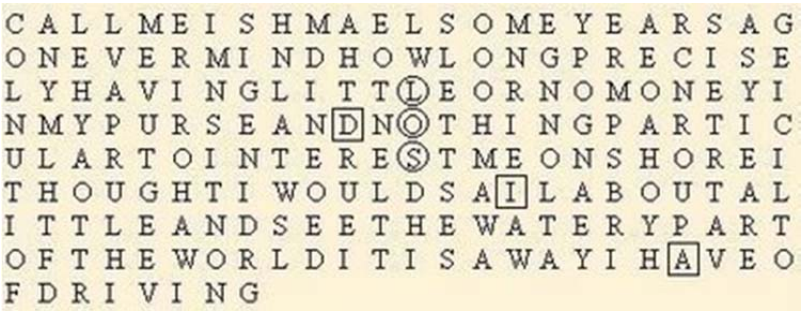
C A L L ME I S H MA E L S O ME Y E A R S A G
O N E V E R MI N D H O WL O N G P R E C I S E
L Y H A V I N G L I T T (L) E O R N O MO N E Y I
N MY P U R S E A N D N (O) T H I N G P A R T I C
U L A R T O I N T E R E (S) T ME O N S H O R E I
T H O U G H T I WO U L D S A I L A B O U T A L
I T T L E A N D S E E T H E WA T E R Y P A R T
O F T H E WO R L D I T I S A WA Y I H A V E O
F D R I V I N G

Para explicar el procedimiento, el algoritmo comienza por la primer letra “s” que existe en el texto y luego busca la primer letra “o” siguiente a la “s”; así se fijó la distancia que debe haber para las

demás letras de la palabra. En esta matriz, desde la letra “s” hasta la siguiente letra “o” hay 24 letras, entonces la distancia entre la “o” y la “l” debe ser de 24 letras.
La distancia entre letras puede llegar hasta números muy elevados como por ejemplo “Itzhak Rabín” que aparece una sola vez en la Torá con un salto de 5000 letras. Y esto no es nada, hay matrices que se han encontrado con un número superior de saltos de letras.
Claro está que este trabajo se realiza millones de veces más rápido de lo que se podría hacer a mano. Es por ello que se requiere poder de còmputo para la búsqueda de una palabra en forma de secuencia de letras equidistantes.
Una vez encontrados todos los resultados de la palabra buscada, el algoritmo muestra de manera bidimensional en la pantalla para ser más apreciable el resultado. Para lo cual, se dividen la longitud total del texto completo en el cual se realizó la búsqueda por la distancia que hay entre las letras de la palabra. En el persente caso, el algoritmo ha encontrado en las primeras 200 letras de "Moby Dick" la palabra “sol” con un salto de 24 letras, entonces el algorirtmo dividió 200 por 24 dando un resultado de 8,333 (8 filas de 24 letras y una de 8 letras).
En la matriz anterior hay 8 filas de 24 letras cada una y la novena fila sólo consta de la tercera parte de 24 (8 letras) para que pueda verse la palabra como si se leyera de arriba a abajo o viceversa. El Algoritmo sólo muestra el centro de la matriz.

1.3.2. Ejemplo Nº2:

Habiendo ingresado la palabra o expresión principal, se introducen nuevas palabras para buscar “alrededor” de la primera. Por ejemplo, se encuentra la palabra "día"con un salto de 53 letras:



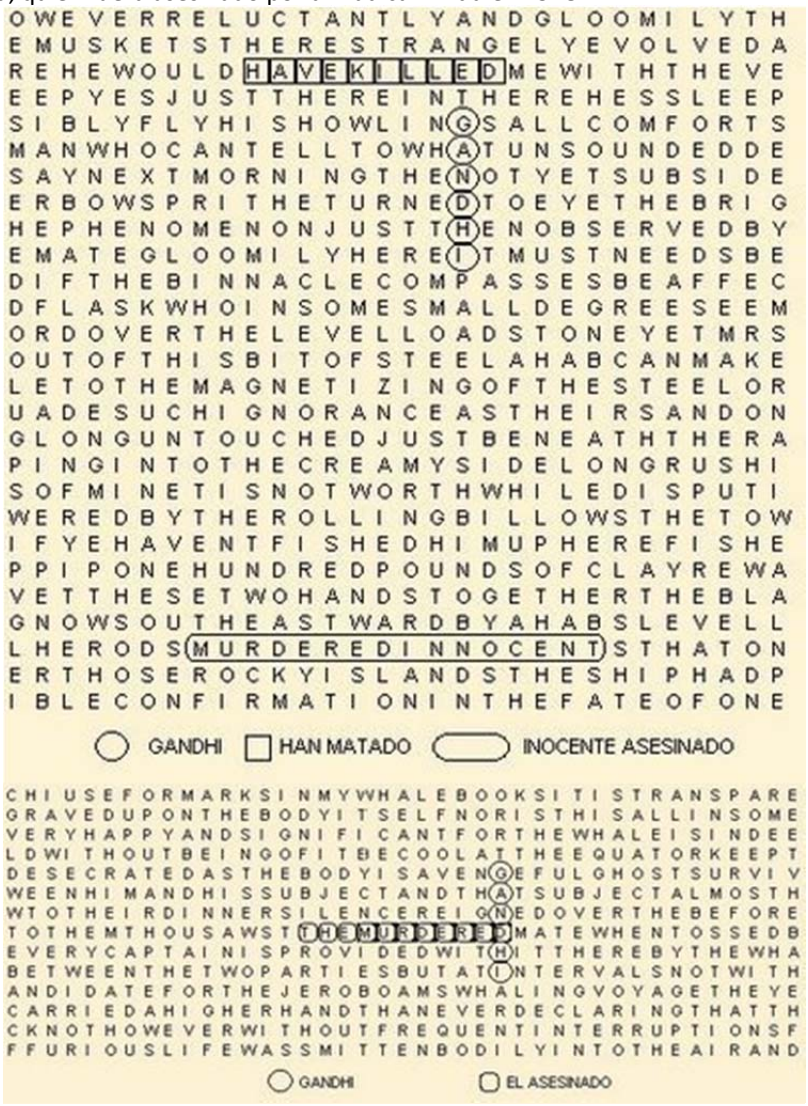
Todo esto se realizó a modo de ejemplo ya que el alegado fenómeno del "código" sólo es buscado en los libros de la versión Koren de la Biblia hebrea.

1.3.3. Ejemplo Nº3:

En una entrevista para la revista norteamericana "Newsweek", Drosnin aseguró“Cuando mis críticos encuentren un mensaje acerca del asesinato de un primer ministro codificado en **Moby Dick**, les creeré”.

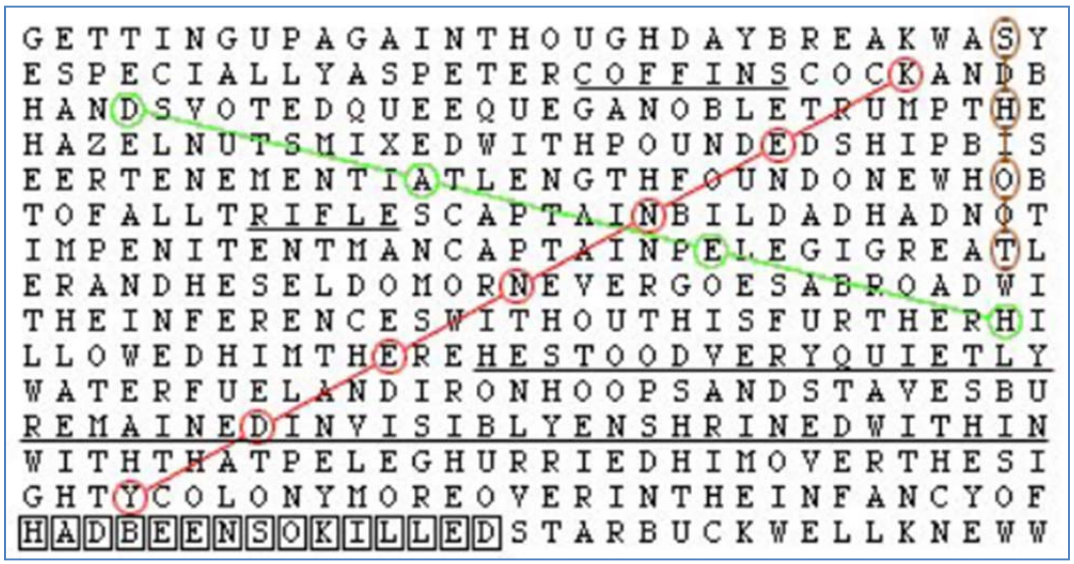
Esto llevó a Brendan McKay a buscar diferentes nombres de personajes asesinados de la historia contemporánea en la versión inglesa de Moby Dick. El resultado fue asombroso, no sólo se encontró con el asesinato de Indira Ghandi, sino también los asesinatos de Martin Luther King, John F. Kennedy, Abraham Lincoln, e Yitzhak Rabin, así como la muerte de Diana, Princesa de Gales.

Por mi parte, señalaba McKay, he encontrado estas matrices en Moby Dick referidas a Mahatma Gandhi, político, pacifista, quien fuera asesinado por un radical hindú en 1948.



1.3.4. Ejemplo Nº4:

Aplicación del método de las secuencias equidistantes, a la novela de "Moby Dick":



Como puede apreciarse, allí aparecen las palabras y frases siguientes:

> Kennedy - had been so killed - shoot - head - remained invisibly enshrined within - he stood very quietly - coffins

La exégesis es bastante simple:

> Kennedy ha sido asesinado, baleado en la cabeza por un asesino que esperaba silenciosamente en un lugar oculto.

2. ALCANCES

2. 1. OBJETIVO

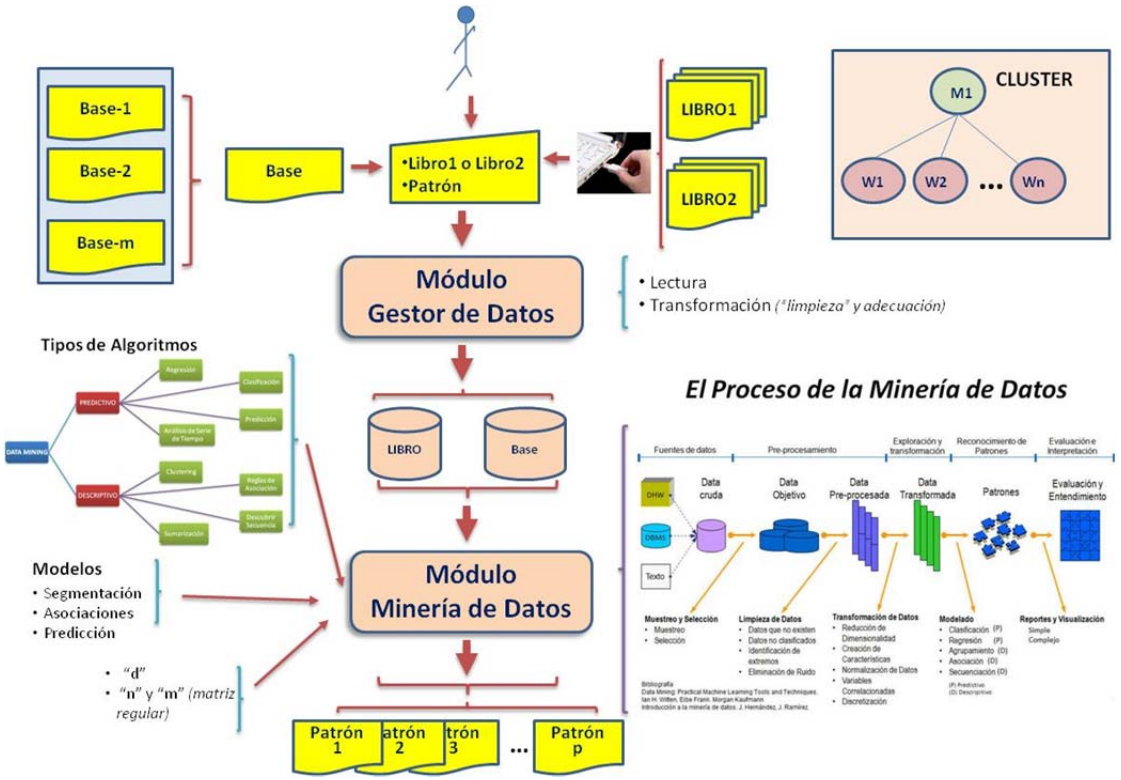
Implementar una solución computacional, con base en una plataforma paralela, que analice patrones en una base de datos de texto, considerando como modelo la búsqueda de patrones implementado en el "Código de la Biblia".

Para lograr el objetivo anterior, considerar los siguientes objetivos específicos:

1. Permitir a la aplicación manejar cualquier texto y además, con potencial de tratamiento para "*n*" componentes distintos *(con un máximo de 10 componentes de una secuencia a buscar, que debe ser tratado como variable) junto a parámetros "d", "m" y "n" variables (distancia y tamaño matriz regular).*
2. Enfocar la aplicación a realizar para usuarios de tipo "Dummies", es decir, usuarios sin ningún conocimiento previo de los temas tratados en la construcción de la solución.
3. Generar una plataforma Web, por medio de la cual se "manejará" la solución paralela, la cual debe caracterizarse por ser "amigable e intuitiva" para el usuario.

2. 2. MODELO GENERAL:

El modelo general de procesamiento comprende dos módulos de gestión: Gestor de Datos *(que incluye un "formateador de texto" capaz de transformar los documentos de nivel "usuario" en archivos utilizables por el módulo de gestión de Minería de Datos)* y Gestor de Minería de Datos, tal como se representa en el diagrama siguiente, en un cluster con un nodo maestro (M1) y "n" nodos esclavos (Wj; j=1,n).



2. 3. CONSIDERACIONES GENERALES

i. Conformar un único grupo de trabajo *("Grupo Curso")* para el modelamiento, diseño y solución del problema planteado, construyendo una solución secuencial y una solución paralela. La primera, no necesariamente en una plataforma paralela, pero debe constituirse en la mejor solución secuencial, lo cual significará su optimización. La segunda, construida e implementada en la plataforma paralela seleccionada.

ii. Considerar dos o más libros como datos *(debe considerarse como opciones de procesamiento en la plataforma paralela y tratamiento de ingreso vía Web).*

iii. El proyecto deberá considerar al menos los siguientes procesos: **Lectura de Libro** en formato "cuasi" original *(pdf o doc, que debe ser un parámetro obtenido vía Web),* ***Transformación*** *de Libro en formato texto, eliminando los espacios entre caracteres y líneas en blanco).* Considerar que el dato *(archivo texto),* comienza con el primer carácter de lectura del tema, no considerando tapas, contratapas, tablas de contenido, índice, prefacio, introducción o reseñas, agradecimientos.

iv. Los resultados del proceso computacional, deben entregarse hacia un informe tipo "pdf" y, también, hacia pantalla vía Web *(como parte de la interfase Web).*

v. Los algoritmos de búsqueda y procesamiento de patrones, deben estar detalladamente fundamentados y descritos en el Capítulo "Fundamentos Teóricos".

vi. Como resultado del procesamiento, debe contemplarse, a lo menos: estadísticas del proceso de búsqueda, que considere: N° de caracteres procesados, N° de patrones a procesar vs N° de patrones encontrados, ubicación de los patrones encontrados *(posición de cada carácter en la cadena de caracteres, página-fila y posición en la fila del carácter),* N° de patrones encontrados, % de éxito de cada patrón, etc.

vii. Elaborar estudios de desempeño *(teóricos y operacionales)* sobre cada solución y optimizarlas.

3. EVALUACIÓN

1. El grupo "curso", a través del "grupo Líder", presentará un primer informe técnico impreso, el día Viernes 16 de mayo de 2014, hasta las 12:00., con los alcances de su proyecto, en cuanto a modelos, técnicas y procesos que se emplearán para dar solución al problema planteado *(Marco Teórico, Diseño de la solución, distribución del trabajo, por grupos integrantes y personas junto con la respectiva cuantificación del esfuerzo).*
2. El grupo "curso", a través del "grupo Líder", presentará un informe de tipo profesional del proyecto, el día Viernes 20 de Junio, hasta las 12:00 hrs, con todo el planteamiento, desarrollo y solución del problema *(alcances de la solución implementada, en cuanto al modelo teórico, técnicas, algoritmos y requerimientos físicos para la habilitación computacional de la plataforma paralela)* y la evaluación del trabajo de cada integrante del "grupo Curso" (esto {ultimo de responsabilidad del "Lider" del proyecto. Adjuntar un CD con los códigos ejecutables, documentos y herramientas de apoyo complementario e informes digitalizados en formato fuente.
3. Como parte del análisis, debe medirse el desempeño de las soluciones tanto secuencia co mno paralela. Así, como parte del análisis e implementación de la solución, debe contemplarse:
   a) Análisis del algoritmo en ambiente secuencial *(Orden, tiempo de ejecución, gráfica de comportamiento de ejecución, análisis final).*
   b) Análisis del algoritmo en ambiente paralelo (idem, más métricas de desempeño paralelo).
   c) Implementación de la solución en "Cluster" con plataforma homogénea de hasta 15 nodos, y, evaluar el efecto de implementar una solución con plataforma heterogénea, incorporando los Pc's P4, dando solución al problema enunciado, con el alcance Web de implementación.
4. El informe Técnico final deberá contener como contenido mínimo:
   1. Resumen (con Abstract *en Ingles)*
   2. Introducción
   3. Marco Teórico
   4. Metodología de Desarrollo
   5. Diseño y Arquitectura del Software
   6. Herramientas Utilizadas
   7. Medidas de Rendimiento
   8. Comparación secuencial vs paralelo.
   9. Carta Gantt *(desde inicio hasta entrega del trabajo)*
   10. *Métricas de esfuerzo (desde el punto de vista de la Ing. de Sw) y su cuantificación.*
   11. Informe de Evaluación de cada integrantes del "Grupo Curso".
   12. Conclusiones y Trabajo Futuro

4. DESARROLLO DEL MARCO TEORICO

El marco teórico de la Fase II, es la sección del proyecto contiene todos los antecedentes teóricos necesarios para planificar, modelar, diseñar e implantar el proceso búsqueda de patrones vía software, en términos de contener "qué es", "cuáles son los procesos", "teoría matemática subyacente", "técnicas y herramientas disponibles"; etc.

Esta parte debe ser 100% de tipo técnica y en el contexto de investigación.

5. BILIOGRAFIA BÁSICA

1. "Breve Historia del Código de la Biblia": http://lcsilva.sbhac.net/Articulos/20.Breve_historia_del_CODIGO_DE_LA_BIBLIA.pdf *(ver "El método de las secuencias equidistantes")*
2. [Mckay et al (1999)]. "Solving The Bible Code Puzzle", Brendan Mckay, Dror Bar-Natan, Maya Bar-Hillel, And Gil Kalai. June 1999, Statistical Science. http://www.math.toronto.edu/~drorbn/Codes/StatSci.pdf
3. [Heughes(2007)] Hueghus, Víctor (2007), "Mineria Web de Uso y Perfiles de Usuario: Aplicaciones con Lógica Difusa". Tesis doctoral, Univ. de Granada, España.
4. [WIK (25-05-2014)] Código de la Biblia. http://es.wikipedia.org/wiki/C%C3%B3digo_de_la_Biblia (acceso 25-Marzo-2014).
5. [Drosnin (2011)] Drosnin, Michael. "El Código Secreto de la Biblia III. La Predicción Final". Ed. Planeta S.A.
6. [Berea (2014)] Algunos Antecedentes sobre "EL CÓDIGO SECRETO DE LA BIBLIA". ESTUDIO Nº 54. Ministerio BÉIT MILÁH, Academia Bíblica BEREA Argentina Estudios Bíblicos. http://www.academiaberea.com/estudios/Codigo-Secreto.pdf (acceso 25-Marzo-2014).
7. [Ninsord (2014)] Ninsord (2014). "El Código de la Biblia. ¿Ficción o realidad?" http://elcodigodelabiblia.blogspot.com/ (acceso 25-Marzo-2014).
8. [Bae89] Ricardo A. Baeza-Yates. Efficient Text Searching. Research Report CS-89-17 (1989), Departament of Computer Science, University of Waterloo, Ontario.
9. [Dav86] Davies, G. y Bowsher, S.. Algorithms for Pattern Matching. Software – Practice and Experience 16 (6), 575–601 (Junio 1986).
10. [Hum91] Hume, A. y Sunday, D. M.. Fast String Searching. Software – Practice and Experience 21 (11), 1221–1248 (Noviembre 1991).
11. [Wat92] Watson, B. W. y Zwaan, G.. A taxonomy of keyword pattern matching algorithms. Computing Science Notes 92/27 (Diciembre 1992), Eindhoven University of Technology, Eindhoven, Holanda.
12. [Wat94] Bruce W. Watson. The performance of single-keyword and multiple-keyword pattern matching algorithms. Computing Science Notes 94/19 (Mayo 1994), Eindhoven University of Technology, Eindhoven, Holanda.
13. [Wat95] Watson, B.W.y Zwaan, G.. Ataxonomy of sublinear multiple keyword pattern matching algorithms. Computing Science Notes 95/ (Abril 1995), Eindhoven University of Technology, Eindhoven, Holanda.

6. ANEXOS

Anexo 6.1. Ejemplo de algoritmos utilizados en Data Mining

Lista de algoritmos de importante utilización e influyentes en tópicos de clasificación.

1. C4.5. Este algoritmo genera clasificadores expresados en términos de árboles de decisión.
2. El algoritmo de *k*-medias. Es un método simple iterativo que particiona un conjunto de datos en un número pre-especificado de conglomerados.
3. SVM (*Support Vector Machine*). Mediante el aprendizaje, este algoritmo trata de encontrar la mejor función de clasificación para distinguir en miembros de distintas clases.
4. El algoritmo *a priori*. Este método encuentra conjuntos de ítems frecuentes usando generación candidata.
5. El algoritmo EM. Es utilizado para clasificar datos de naturaleza continua y para estimar su correspondiente función de densidad.
6. PageRank. Es un algoritmo de búsqueda sobre hipervínculos en la web. Gracias a este método es que Google funciona.
7. AdaBoost. Emplea métodos que utilizan múltiples *"learners"* para resolver un problema.
8. *k*NN. Memoriza el conjunto de datos de entrenamiento y realiza una clasificación sólo si los atributos del objeto de prueba coinciden exactamente con los ejemplos del entrenamiento.
9. Bayes ingenuo (*Naive Bayes*). Dado un conjunto de objetos, que pertenecen a una clase conocida, construye una regla que permite asignar objetos futuros a una clase.
10. CART (*Classification and Regression Trees*). Se trata de un procedimiento recursivo de partición capaz de procesar atributos nominales y continuos como objetivos o predictores.

Anexo 6.2. Algoritmos de minería de datos (Analysis Services: Minería de datos: Microsoft) – Antecedentes de apoyo al Marco teórico

http://msdn.microsoft.com/en-us/library/ms175595(v=sql.120).aspx

Un *algoritmo de minería de datos* es un conjunto de cálculos y reglas heurísticas que permite crear un modelo de minería de datos a partir de los datos. Para crear un modelo, el algoritmo analiza primero los datos proporcionados, en busca de tipos específicos de patrones o tendencias. El algoritmo usa los resultados de este análisis para definir los parámetros óptimos para la creación del modelo de minería de datos. A continuación, estos parámetros se aplican en todo el conjunto de datos para extraer patrones procesables y estadísticas detalladas.
El modelo de minería de datos que crea un algoritmo a partir de los datos puede tomar diversas formas, incluyendo:

- Un conjunto de clústeres que describe cómo se relacionan los casos de un conjunto de datos.
- Un árbol de decisión que predice un resultado y que describe cómo afectan a este los distintos criterios.
- Un modelo matemático que predice las ventas.
- Un conjunto de reglas que describen cómo se agrupan los productos en una transacción, y las probabilidades de que dichos productos se adquieran juntos.

Analysis Services *(de Microsoft SQL Server)* proporciona varios algoritmos que puede usar en las soluciones de minería de datos. Estos algoritmos son implementaciones de algunas de las metodologías más conocidas usadas en la minería de datos. Todos los algoritmos de minería de datos de Microsoft se pueden personalizar y son totalmente programables, bien mediante las API proporcionadas o bien mediante los componentes de minería de datos de SQL Server Integration Services.

También puede usar algoritmos de minería de datos desarrollados por terceros que cumplan la especificación OLE DB para minería de datos, o desarrollar algoritmos personalizados que se pueden registrar como servicios para usarlos a continuación en el marco de la minería de datos de SQL Server.

Elegir el algoritmo correcto

La elección del mejor algoritmo para una tarea analítica específica puede ser un desafío. Aunque puede usar diferentes algoritmos para realizar la misma tarea, cada uno de ellos genera un resultado diferente, y algunos pueden generar más de un tipo de resultado. Por ejemplo, puede usar el algoritmo Árboles de decisión de Microsoft no solo para la predicción, sino también como una forma de reducir el número de columnas de un conjunto de datos, ya que el árbol de decisión puede identificar las columnas que no afectan al modelo de minería de datos final.

Elegir un algoritmo por tipo

Analysis Services incluye los siguientes tipos de algoritmos:

- **Algoritmos de clasificación**, que predicen una o más variables discretas, basándose en otros atributos del conjunto de datos.
- **Algoritmos de regresión**, que predicen una o más variables continuas, como las pérdidas o los beneficios, basándose en otros atributos del conjunto de datos.
- **Algoritmos de segmentación**, que dividen los datos en grupos, o clústeres, de elementos que tienen propiedades similares.
- **Algoritmos de asociación**, que buscan correlaciones entre diferentes atributos de un conjunto de datos. La aplicación más común de esta clase de algoritmo es la

creación de reglas de asociación, que pueden usarse en un análisis de la cesta de compra.

- **Algoritmos de análisis de secuencias**, que resumen secuencias o episodios frecuentes en los datos, como un flujo de rutas web.

Sin embargo, no hay ninguna razón por la que deba limitarse a un algoritmo en sus soluciones. Los analistas experimentados usarán a veces un algoritmo para determinar las entradas más eficaces (es decir, variables) y luego aplicarán un algoritmo diferente para predecir un resultado concreto basado en esos datos. La minería de datos de SQL Server le permite generar varios modelos en una única estructura de minería de datos, por lo que en una solución de minería de datos puede usar un algoritmo de clústeres, un modelo de árboles de decisión y un modelo de Bayes naïve para obtener distintas vistas de los datos. También puede usar varios algoritmos dentro de una única solución para realizar tareas independientes: por ejemplo, podría usar la regresión para obtener predicciones financieras, y un algoritmo de red neuronal para realizar un análisis de los factores que influyen en las ventas.

Elegir un algoritmo por tarea

Con el fin de ayudar a seleccionar un algoritmo para el uso con una tarea específica, la tabla siguiente proporciona sugerencias para los tipos de tareas para las que se usa normalmente cada algoritmo.

EJEMPLOS DE TAREAS	ALGORITMOS DE MICROSOFT QUE SE PUEDEN USAR
Predecir un atributo discreto • Marcar los clientes de una lista de posibles compradores como clientes con buenas o malas perspectivas. • Calcular la probabilidad de que un servidor genere un error en los próximos 6 meses. • Clasificar la evolución de los pacientes y explorar los factores relacionados.	Algoritmo de árboles de decisión de Microsoft Algoritmo Bayes naive de Microsoft Algoritmo de clústeres de Microsoft Algoritmo de red neuronal de Microsoft
Predecir un atributo continuo • Pronosticar las ventas del año próximo. • Predecir los visitantes del sitio a partir de tendencias históricas y estacionales proporcionadas. • Generar una puntuación de riesgo a partir de datos demográficos.	Algoritmo de árboles de decisión de Microsoft Algoritmo de serie temporal de Microsoft Algoritmo de regresión lineal de Microsoft
Predecir una secuencia • Realizar un análisis clickstream del sitio web de una empresa. • Analizar los factores que dan como resultado errores en el servidor. • Capturar y analizar secuencias de actividades durante las visitas de pacientes externos, para formular las prácticas recomendadas en las actividades comunes.	Algoritmo de clústeres de secuencia de Microsoft
Buscar grupos de elementos comunes en las transacciones • Usar el análisis de la cesta de la compra para determinar la posición del producto. • Sugerir a un cliente la compra de productos adicionales. • Analizar los datos de una encuesta a los visitantes a un evento, para descubrir qué actividades o stands estaban correlacionados con el fin de programar actividades futuras.	Algoritmo de asociación de Microsoft Algoritmo de árboles de decisión de Microsoft
Buscar grupos de elementos similares • Crear grupos de pacientes con perfiles de riesgo en función de atributos como datos demográficos y comportamientos. • Analizar usuarios mediante patrones de búsqueda y compra de productos. • Identificar servidores con características de uso similares.	Algoritmo de clústeres de Microsoft Algoritmo de clústeres de secuencia de Microsoft

Contenido relacionado *(Fuente: http://msdn.microsoft.com/en-us/library/ms175595(v=sql.120).aspx)*

En la tabla siguiente se incluyen vínculos a recursos de aprendizaje para cada uno de los algoritmos de minería de datos que se proporcionan en Analysis Services:

Descripción básica del algoritmo	**Explica lo que hace que el algoritmo y cómo funciona, y describe los posibles escenarios empresariales donde podría resultar útil.**
	Algoritmo de asociación de Microsoft Algoritmo de clústeres de Microsoft Algoritmo de árboles de decisión de Microsoft Algoritmo de regresión lineal de Microsoft Algoritmo de regresión logística de Microsoft Algoritmo Bayes naive de Microsoft Algoritmo de red neuronal de Microsoft Algoritmo de clústeres de secuencia de Microsoft Algoritmo de serie temporal de Microsoft
Referencia técnica	Proporciona detalles técnicos sobre la implementación del algoritmo, con referencias académicas según sea necesario. Muestra los parámetros que pueden establecerse para controlar el comportamiento del algoritmo y personalizar los resultados en el modelo. Describe los requisitos de los datos y proporciona sugerencias de rendimiento si es posible.
	Referencia técnica del algoritmo de asociación de Microsoft Referencia técnica del algoritmo de clústeres de Microsoft Referencia técnica del algoritmo de árboles de decisión de Microsoft Referencia técnica del algoritmo de regresión lineal de Microsoft Referencia técnica del algoritmo de regresión logística de Microsoft Referencia técnica del algoritmo Bayes naive de Microsoft Referencia técnica del algoritmo de red neuronal de Microsoft Referencia técnica del algoritmo de clústeres de secuencia de Microsoft Referencia técnica del algoritmo de serie temporal de Microsoft
Contenido del modelo	Explica cómo está estructurada la información dentro de cada tipo de modelo de minería de datos, y cómo interpretar la información almacenada en cada uno de los nodos.
	Contenido del modelo de minería de datos para los modelos de asociación (Analysis Services - Minería de datos) Contenido del modelo de minería de datos para los modelos de agrupación en clústeres (Analysis Services - Minería de datos) Contenido del modelo de minería de datos para los modelos de árboles de decisión (Analysis Services - Minería de datos) Contenido del modelo de minería de datos para los modelos de regresión lineal (Analysis Services - Minería de datos) Contenido del modelo de minería de datos para los modelos de regresión logística (Analysis Services - Minería de datos) Contenido del modelo de minería de datos para los modelos Bayes naive (Analysis Services - Minería de datos) Contenido del modelo de minería de datos para los modelos de red neuronal (Analysis Services - Minería de datos) Contenido del modelo de minería de datos para los modelos de agrupación en clústeres de secuencia(Analysis Services - Minería de datos) Contenido del modelo de minería de datos para los modelos de serie temporal (Analysis Services - Minería de datos)
Consultas de minería de datos	Proporciona varias consultas que se pueden usar con cada tipo de modelo. Los ejemplos incluyen consultas de contenido que le proporcionan más información sobre los patrones del modelo, así como consultas de predicción para ayudarle a crear predicciones basadas en esos patrones.

Anexo 6.3. Algoritmos de búsqueda de subcadenas

Fuente: http://es.wikipedia.org/wiki/Algoritmos_de_b%C3%BAsqueda_de_subcadenas

A este tipo de algoritmos también se les llama Algoritmos de patrones en un texto, algoritmos de emparejamiento de secuencias, algoritmos de casamiento de secuencias o simplemente por su nombre en inglés string matching. Este tipo de algoritmos persiguen encontrar subcadena/s con alguna propiedad en una cadena de caracteres.
Índice

Terminología

Normalmente se denomina patrón/es a la/ subcadena/s buscada/s y texto a la cadena en la que se realiza la búsqueda. Se suelen emplear las letras m y n para referirnos a la longitud de un patrón y a la longitud del texto respectivamente. Podemos asumir que n > m

Clasificación

Este tipo de algoritmos se pueden clasificar según el número de subcadenas que se intentan buscar en simples, se busca sólo una subcadena, y múltiples, se buscan varias subcadenas.[1] [2]

Algoritmos de búsqueda simple de subcadenas

También llamados por su denominación en inglés Single string Matching. En este tipo de algoritmos sólo se busca una subcadena a la que llamamos patrón, es decir el objetivo es encontrar todas las ocurrencias del patrón p dentro del texto. Este tipo de algoritmos se suelen agrupar en alguno de los siguientes tipos[3]

1. Fuerza bruta. La idea es ir deslizando el patrón sobre el texto de izquierda a derecha, comparándolo con las subcadenas del mismo tamaño que empiezan en cada carácter del texto.
2. Leer todos los caracteres del texto uno a uno modificando en cada paso algunas variables que permitan identificar posibles ocurrencias. A este tipo pertenecen los algoritmos de Knuth-Morris-Pratt,[4] Shift-Or[5] o búsqueda simple con autómata determinista.
3. Buscar el patrón en una ventana que se desliza a lo largo del texto. Para cada posición de esta ventana buscamos de derecha a izquierda un sufijo de la ventana que corresponda a un sufijo del patrón. A este tipo pertenecen los algoritmos deBoyer-Moore,[6] Boyer-Moore-Haspool[7] y Sunday Quick Search.[8] Este tipo de algoritmos no suelen funcionar bien cuando el tamaño del patrón el pequeño y hay una probabilidad alta de encontrarlo en el texto.
4. La búsqueda se realiza de derecha a izquierda dentro de una ventana, pero en este esquema se busca el sufijo más largo en la ventana que es subcadena del patrón. Ejemplos de este tipo de algoritmos son BDM,[9] BNDM[10] y BOM.[11] Este tipo de algoritmos para patrones pequeños no suelen funcionar bien.
5. Esquemas basados en funciones hash. Ejemplo de este tipo de algoritmos es el de Karp-Rabin.[12]

Algoritmos de búsqueda múltiple de subcadenas

También llamados por su denominación en inglés Multiple String Matching. Ahora no tenemos un sólo patron p a buscar sino que contamos con un conjunto P={p1,...,pl} de patrones. La solución que se suele adoptar es la extensión de los esquemas anteriores para el caso múltiple. Por tanto tenemos los siguientes subtipos:

1. Fuerza bruta
2. Extensión del tipo 2 de algoritmos de búsqueda simple de subcadenas. De este tipo de algoritmos son los de Aho-Corasick,13 Multiple Shift-And y búsqueda múltiple con autómata determinista.
3. Extensión del tipo 3 de algoritmos de búsqueda simple de subcadenas. De este tipo son los algoritmos de Commentz-Walter,14 Set Horspool, Wu-Manber.15
4. Extensión del tipo 4 de algoritmos de búsqueda simple de subcadenas. De este tipo son los algoritmos SBOM, Multiple BNDM,16 DAWG-MATCH.17
5. Extensión del tipo 5 de algoritmos de búsqueda simple de subcadenas

Referencias

1. Román Roset Mayals,Diseño de una aplicación bioinformática para el estudio de repeticiones de patrones en cadenas de DNA. Memoria 2003
2. Sergio Talens-Oliag Análisis de algoritmos de búsqueda de un solo patrón. Proyecto Fin de Carrera 1997. U Politécnica de Valencia
3. Román Roset Mayals,Diseño de una aplicación bioinformática para el estudio de repeticiones de patrones en cadenas de DNA. Memoria 2003
4. D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. SIAM J.Comput, 6(2):323–350, 1977
5. R. Baeza-Yates and G. Gonnet. A new approach to text searching. Comm. ACM, 35(10):74–82, 1992
6. R. S. Boyer and J. S. Moore. A fast string searching algorithm. Comm. ACM, 20(10):762–772, 1977
7. R. Horspool. Practical fast searching in strings. Softw. Pract. Exp.,10(6):501–506, 1980
8. Daniel M. Sunday. A Very Fast Substring Search Algorithm. Comunications of the ACM 33 (8),132–142 (Agosto 1990)
9. A. Czumaj, M. Crochemore, L. Gasieniec, S. Jarominek, W. Plandowski, T. Lecroq, and W. Rytter. Speeding up two string-matching algorithms. Algorithmica, 12:247–267, 1994.
10. G. Navarro and M. Raffinot. A bit-parallel approach to suffix automata:Fast extended string matching. In Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching, number 1448 in Lecture Notes in Computer Science, pages 14–33. Springer-Verlag, Berlin, 1998
11. Cyril Allauzen, Maxime Crochemore, and Mathieu Raffinot. Factor oracle:A new structure for pattern matching. In Conference on Current Trends in Theory and Practice of Informatics, pages 295–310, 1999.
12. Karp, R. M. y Rabin, M. O.. Efficient Randomized Pattern Matching Algorithms. IBM J. Res.Develop.31 (2), 249–260 (1987)
13. A. V. Aho and M. Corasick. Efficient string matching: An aid to bibliographic search. Comm. ACM, 18(6):333–340, 1975
14. B. Commentz-Walter. A string matching algorithm fast on the average. In 6th, number 71 in Lecture Notes in Computer Science, pages 118–132. H.A.Maurer, 1979
15. S. Wu and U. Manber. A fast algorithm for multi-pattern searching. Technical Report TR-94-17, Chung-Cheng University, 1994
16. M. Crochemore and W.Rytter. Text Algorithms. Oxford University Press,1994.
17. Maxime Crochemore, Artur Czumaj, Leszek Gasieniec, Thierry Lecroq,Wojciech Plandowski, and Wojciech Rytter. Fast practical multi-pattern matching. Information Processing Letters, 71(3-4):107–113, 1999.