Clustering de Documentos Web basado en una Matriz de Términos Frecuentes por Oraciones de Documentos, FP-Growth y Bisecting k-means

ANEXOS

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz

Director: Ph.D. (c) MSc Carlos Alberto Cobos Lozada

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información
Gestión de la Información – Búsqueda Web
Popayán, Abril de 2011

TABLA DE CONTENIDO

ANEXO A – LISTA DE STOPWORDS 5

1 LISTA DE STOPWORDS 6

2 LISTA DE VERBOS 7

ANEXO B – ALGORITMO DE PORTER 9

3 ALGORITMO DE PORTER 10

3.1 Introducción 10

3.2 Algoritmo 11

ANEXO C – ALGORITMO FPGROWTH 16

4 FPGROWTH 17

4.1 Introducción 17

4.2 Pasos del Algoritmo 18

ANEXO D – IMPLEMENTACION DE LUCENE.NET 20

5 LUCENE.NET 21

ANEXO E – CASOS DE USO REALES 23

6 CASOS DE USO REALES 24

ANEXO F – DIAGRAMA DE CLASES DEL SISTEMA 25

7 DIAGRAMA DE CLASES 26

8 CLASES DEL SISTEMA 28

ANEXO G – SURVEY 35

REFERENCIAS 48

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz
Ph. D. Carlos Alberto Cobos Lozada

LISTA DE FIGURAS

Figura 1. Diagrama general de clases
Figura 2. Vista Detallada de la clase Bisecting Kmeans
Figura 3. Vista Detallada de la clase ConstantesLucene
Figura 4. Vista Detallada de la clase FPGrowth
Figura 5. Vista Detallada de la clase GrupoHSKmeans
Figura 6. Vista Detallada de la clase HSKmeans
Figura 7. Vista Detallada de la clase HSKmeansFila
Figura 8. Vista Detallada de la clase Indice
Figura 9. Vista Detallada de la clase Kmeans
Figura 10. Vista Detallada de la clase MatrizFD
Figura 11. Vista Detallada de la clase MatrizFFD
Figura 12. Vista Detallada de la clase MatrizFD
Figura 13. Vista Detallada de la clase MatrizKmeans
Figura 14. Vista Detallada de la clase MatrizResultados
Figura 15. Vista Detallada de la clase MatrizTFD

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz
Ph. D. Carlos Alberto Cobos Lozada

LISTA DE TABLAS

Tabla 1. Caso de uso real Iniciar Sesion
Tabla 2. Caso de uso real Calificar grupos y documentos
Tabla 3. Descripción de las clases

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz
Ph. D. Carlos Alberto Cobos Lozada

ANEXO A – LISTA DE STOPWORDS Y VERBOS EN INGLÉS

1 LISTA DE STOPWORDS

A continuación se presenta la lista de Stop Words que se utilizó en la fase de pre-procesamiento del algoritmo WDC-PTFBK, la cual se tomó del sitio de la Universidad UNINE, de Chile, específicamente de la página web "IR Multilingual Resources at UniNE" (http://www.unine.ch/jacques.savoy/clef/index.html).

English StopWords (palabras en Inglés).

"a", "a's", "able", "about", "above", "according", "accordingly", "across", "actually", "after", "afterwards", "again", "against", "ain't", "all", "allow", "allows", "almost", "alone", "along", "already", "also", "although", "always", "am", "among", "amongst", "an", "and", "another", "any", "anybody", "anyhow", "anyone", "anything", "anyway", "anyways", "anywhere", "apart", "appear", "appreciate", "appropriate", "are", "aren't", "around", "as", "aside", "ask", "asking", "associated", "at", "available", "away", "awfully", "b", "be", "became", "because", "become", "becomes", "becoming", "been", "before", "beforehand", "behind", "being", "believe", "below", "beside", "besides", "best", "better", "between", "beyond", "both", "brief", "but", "by", "c", "c'mon", "c's", "came", "can", "can't", "cannot", "cant", "cause", "causes", "certain", "certainly", "changes", "clearly", "co", "com", "come", "comes", "concerning", "consequently", "consider", "considering", "contain", "containing", "contains", "corresponding", "could", "couldn't", "course", "currently", "d", "definitely", "described", "despite", "did", "didn't", "different", "do", "does", "doesn't", "doing", "don't", "done", "down", "downwards", "during", "e", "each", "edu", "eg", "eight", "either", "else", "elsewhere", "enough", "entirely", "especially", "et", "etc", "even", "ever", "every", "everybody", "everyone", "everything", "everywhere", "ex", "exactly", "example", "except", "f", "far", "few", "fifth", "first", "five", "followed", "following", "follows", "for", "former", "formerly", "forth", "four", "from", "further", "furthermore", "g", "get", "gets", "getting", "given", "gives", "go", "goes", "going", "gone", "got", "gotten", "greetings", "h", "had", "hadn't", "happens", "hardly", "has", "hasn't", "have", "haven't", "having", "he", "he's", "hello", "help", "hence", "her", "here", "here's", "hereafter", "hereby", "herein", "hereupon", "hers", "herself", "hi", "him", "himself", "his", "hither", "hopefully", "how", "howbeit", "however", "i", "i'd", "i'll", "i'm", "i've", "ie", "if", "ignored", "immediate", "in", "inasmuch", "inc", "indeed", "indicate", "indicated", "indicates", "inner", "insofar", "instead", "into", "inward", "is", "isn't", "it", "it'd", "it'll", "it's", "its", "itself", "j", "just", "k", "keep", "keeps", "kept", "know", "knows", "known", "l", "last", "lately", "later", "latter", "latterly", "least", "less", "lest", "let", "let's", "like", "liked", "likely", "little", "look", "looking", "looks", "ltd", "m", "mainly", "many", "may", "maybe", "me", "mean", "meanwhile", "merely", "might", "more", "moreover", "most", "mostly", "much", "must", "my", "myself", "n", "name", "namely", "nd", "near", "nearly", "necessary", "need", "needs", "neither", "never", "nevertheless", "new", "next", "nine", "no", "nobody", "non", "none", "noone", "nor", "normally", "not", "nothing", "novel", "now", "nowhere", "o", "obviously", "of", "off", "often", "oh", "ok", "okay", "old", "on", "once", "one", "ones", "only", "onto", "or", "other", "others", "otherwise", "ought", "our", "ours", "ourselves", "out", "outside", "over", "overall", "own", "p", "particular", "particularly", "per", "perhaps", "placed", "please", "plus", "possible", "presumably", "probably", "provides", "q", "que", "quite", "qv", "r", "rather", "rd", "re", "really", "reasonably", "regarding", "regardless", "regards",

"relatively", "respectively", "right", "s", "said", "same", "saw", "say", "saying", "says", "second", "secondly", "see", "seeing", "seem", "seemed", "seeming", "seems", "seen", "self", "selves", "sensible", "sent", "serious", "seriously", "seven", "several", "shall", "she", "should", "shouldn't", "since", "six", "so", "some", "somebody", "somehow", "someone", "something", "sometime", "sometimes", "somewhat", "somewhere", "soon", "sorry", "specified", "specify", "specifying", "still", "sub", "such", "sup", "sure", "t", "t's", "take", "taken", "tell", "tends", "th", "than", "thank", "thanks", "thanx", "that", "that's", "thats", "the", "their", "theirs", "them", "themselves", "then", "thence", "there", "there's", "thereafter", "thereby", "therefore", "therein", "theres", "thereupon", "these", "they", "they'd", "they'll", "they're", "they've", "think", "third", "this", "thorough", "thoroughly", "those", "though", "three", "through", "throughout", "thru", "thus", "to", "together", "too", "took", "toward", "towards", "tried", "tries", "truly", "try", "trying", "twice", "two", "u", "un", "under", "unfortunately", "unless", "unlikely", "until", "unto", "up", "upon", "us", "use", "used", "useful", "uses", "using", "usually", "uucp", "v", "value", "various", "very", "via", "viz", "vs", "w", "want", "wants", "was", "wasn't", "way", "we", "we'd", "we'll", "we're", "we've", "welcome", "well", "went", "were", "weren't", "what", "what's", "whatever", "when", "whence", "whenever", "where", "where's", "whereafter", "whereas", "whereby", "wherein", "whereupon", "wherever", "whether", "which", "while", "whither", "who", "who's", "whoever", "whole", "whom", "whose", "why", "will", "willing", "wish", "with", "within", "without", "won't", "wonder", "would", "wouldn't", "x", "y", "yes", "yet", "you", "you'd", "you'll", "you're", "you've", "your", "yours", "yourself", "yourselves", "z", "zero"

2 LISTA DE VERBOS

A continuación se presenta la lista de verbos en inglés que se utilizó para complementar la fase de etiquetado del algoritmo WDC-PTFBK, la cual se tomo de los sitios "Vocabulix", específicamente de la página web "Verbos en Inglés" (http://www.vocabulix.com/conjugacion/Verbos-Ingles.html), y también se tuvo en cuenta el sitio "Speakspeak Better English", específicamente de la página web "Lista de verbos irregulares en Inglés / Los participios pasados" (http://www.speakspeak.com/html/d2f_resources_irregulares_verbos_ingles_es.htm)

English Verbs (Verbos en Inglés)

"accompany", "accustom", "act", "add", "address", "advertise", "agree", "aid", "amuse", "annoy", "answer", "appeal", "appear", "approach", "arise", "arrange", "arrest", "arrive", "ask", "assist", "attend", "awake", "balance", "banish", "bark", "bear", "beat", "become", "beg", "begin", "behave", "believe", "belong", "bend", "bet", "bind", "bite", "bless", "blow", "board", "boil", "break", "breathe", "bring", "brush", "build", "burn", "burst", "buy", "call", "care", "carry", "catch", "change", "charge", "check", "choose", "clean", "climb", "cling", "close", "comb", "come", "complete", "consist", "cook", "cost", "count", "cover", "crash", "crawl", "creep", "cross ", "cry", "cut", "dance", "deal", "declare", "delay", "deliver", "deny", "dial", "die", "dig", "dine", "do", "draw", "dress", "drink", "drive", "drop", "dry", "enclose", "engage", "enjoy", "envy", "exclaim", "explain", "express", "fail", "fall", "fasten", "feed", "feel", "fight", "file", "fill", "find", "find out", "finish", "fire", "fish", "fix", "flee", "fly", "follow", "forbid", "foresee", "forget", "forgive", "freeze", "frighten", "fry", "gain", "get", "give", "go", "grind", "grow", "guess", "hang", "happen", "have", "hear", "help", "hide", "hit", "hold", "hope", "hurry", "hurt", "imagine", "iron", "judge", "keep", "kill", "kiss", "know", "laugh", "lay",

"lead", "leak", "lean", "learn", "leave", "lend", "let", "lie", "light", "like", "lock", "look", "lose", "make", "manage", "mark", "marry", "massage", "mean", "measure", "meet", "melt", "milk", "miss", "mistake", "misunderstand", "move", "observe", "offer", "open", "order", "observe", "overcome", "park", "pass", "pay", "perform", "phone", "pick", "plan", "play", "please", "plough", "polish", "pour", "practice", "pray", "prefer", "prepare", "promise", "pronounce", "pull", "punish", "push", "put", "rain", "raise", "reach", "read", "realize", "rebuild", "receive", "refuse", "register", "Remain", "Remember", "Repair", "repeat", "report", "request", "Require", "Reserve", "Resolve", "rest", "Return", "rid", "ride", "ring", "rise", "Row", "run", "Save", "saw", "say", "Search", "see", "seek", "sell", "send", "set", "settle", "shake", "shed", "shine", "shoot", "show", "shrink", "shut", "sign", "sing", "sink", "sit", "sleep", "slide", "slip", "smell", "smile", "smoke", "snow", "speak", "speed", "spend", "spill", "spin", "split", "spoil", "spread", "spring", "stand", "stay", "steal", "stick", "stink", "stop", "stretch", "strike", "study", "suffer", "swallow", "swell", "swim", "swing", "switch", "take", "talk", "teach", "tear", "tell", "thank", "think", "throw", "thrust", "tire", "touch", "train", "trap", "travel", "trouble", "try", "turn", "undergo", "understand", "undertake", "undo", "unpack", "use", "visit", "wait", "wake", "walk", "want", "warm", "warn", "wash", "watch", "water", "wear", "weigh", "whistle", "win", "wind", "wish", "withdraw", "withstand", "work", "wrap up", "wreck", "write"

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz
Ph. D. Carlos Alberto Cobos Lozada

ANEXO B – ALGORITMO DE PORTER

3 ALGORITMO DE PORTER

El Algoritmo de Porter es un proceso para remover los sufijos comunes morfológicos e inflexionales de las palabras en inglés. Se usa como un proceso de normalización que usualmente se realiza para establecer un Sistema de Recuperación. El artículo original del algoritmo (stemming algorithm paper) se escribió en 1979 en el Laboratorio de Información. Actualmente existen algoritmos de Porter especializados a otros lenguajes, entre ellos el español.

3.1 Introducción

La eliminación de sufijos por medios automáticos es una operación que es especialmente útil en el campo de la Recuperación de la Información (IR). En un ambiente típico de Recuperación de Información un documento está representado por un vector de palabras o términos. Los términos con una raíz común, normalmente tienen significado similar, por ejemplo:

CONNECT
CONNECTED
CONNECTING
CONNECTION
CONNECTIONS

Frecuentemente, el desempeño de un sistema de IR se mejorará si los grupos de términos como éstos son combinados dentro de un solo término. Esto se puede hacer por la eliminación de varios sufijos –ED, -ING, -ION, -IONS para dejar solo el término CONNECT. Además, el proceso de eliminación reducirá el número total de términos en el sistema de IR, y por lo tanto reduce el tiempo y la complejidad de los datos en el sistema, lo cual es siempre conveniente.

Muchas estrategias para la eliminación de sufijos se han reportado en la literatura. La naturaleza de la tarea variará considerablemente dependiendo de si se utiliza un diccionario de raíces, o una lista de sufijos, y por supuesto del propósito para el cual el eliminador de sufijos se ha hecho. Asumiendo que no se hace uso de un diccionario de raíces, y que el propósito de la tarea es mejorar el desempeño del sistema de IR, el programa de eliminación de sufijos normalmente dará una lista explícita de sufijos, y, con cada sufijo, el criterio bajo el cual este puede ser eliminado de una palabra para dejar una raíz válida. Los principales aspectos positivos del algoritmo de Porter son: es pequeño, rápido y sencillo.

3.2 Algoritmo

Una consonante será denotada por *c*, una vocal por *v*. Una lista ccc… de longitud más grande que 0 será denotada por C, y una lista vvv… de longitud más grande que 0 será denotada por V. Cualquier palabra, o parte de una palabra, por lo tanto tiene una de las cuatro formas:

CVCV…C
CVCV…V
VCVC…C
VCVC…V

Las cuales pueden ser representadas, a su vez, por una sola forma:

[C] VCVC …[V]

Donde los paréntesis cuadrados denotan presencia arbitraria de sus contenidos. Usando $(VC)^m$ para denotar VC repetido m veces, esto puede de nuevo ser escrito como:

$[C](VC)^m[V]$.

m será llamado la medida de cualquier palabra o parte de una palabra cuando se represente de esta forma. El caso *m* = 0 cubre la palabra nula. Aquí se presentan algunos ejemplos:

m = 0 TR, EE, TREE, Y, BY.
m = 1 TROUBLE, OATS, TREES, IVY.
m = 2 TROUBLES, PRIVATE, OATEN, ORRERY.

Las reglas para la eliminación de sufijos serán dadas en la forma:

(Condición) S1 -> S2

Esto significa que si una palabra finaliza con el sufijo S1, y la raíz antes de S1 satisface la condición dada, S1 es reemplazada por S2. La condición está normalmente dada en términos de m, por ejemplo:

(*m* > 1) EMENT ->

Aquí S1 es “EMENT” y S2 es nulo. Esto convertiría REPLACEMENT a REPLAC, puesto que REPLAC es un parte de la palabra para la que *m*=2. La parte de la condición puede también contener lo siguiente:

*S – la raíz finaliza con S (y similarmente para las otras letras)
v- la raíz contiene una vocal.
*d – la raíz finaliza con una consonante doble (por ejemplo, -TT, -SS).
*o – la raíz finaliza con cvc, donde la segunda c no es W, X, o Y (por ejemplo – WIL, -HOP).

Y la parte de la condición también puede contener expresiones con and, or y not, de modo que:

(*m* > 1 and (*S or *T))
Prueba una raíz con m > 1 que finalice en S o T, mientras,
(*d and not (*L or *S or *Z))

Prueba una raíz que finalice con una consonante doble distinta de L, S o Z. Las condiciones elaboradas como esta son requeridas muy raramente.

En un conjunto de reglas escritas unas tras otras, solo una se cumple, y esta será una con la correspondencia más larga de S1 para una palabra dada. Por ejemplo, con:

SSES -> SS
IES -> I
SS -> SS
S ->

(Aquí las condiciones son todas nulas) CARESSES corresponde a CARESS puesto que SSES es la coincidencia más larga para S1. Igualmente CARESS corresponde a CARESS (S1 = “SS”) y CARES a CARE (S1 = “S”).

El algoritmo presenta cinco pasos, los cuales están compuestos por reglas que se aplican a las palabras de las que se obtendrá la raíz. En estas reglas, los ejemplos de su aplicación (con éxito o no) se presentan a la derecha en minúsculas.

Paso 1a

SSES -> SS	caresses -> caress
IES -> I	ponies ->poni
	Ties -> ti
SS -> SS	caress -> caress
S ->	cats -> cat

Paso 1b

(m > 0) EED -> EE	feed -> feed
	agreed -> agree
(*v*) ED ->	plastered -> plaster
	Bled -> bled (*v*)
ING ->	motoring -> motor
	Sing -> sing

Si la segunda o tercera de las reglas en el Paso 1b es exitoso, se hace lo siguiente:

AT -> ATE	conflat(ed) -> conflate
BL -> BLE	troubl(ing) -> trouble
IZ -> IZE	siz(ed) -> size

(*d and not (*L or *S or *Z)) ->sola letra

hopping (ing) -> hop
Tann(ed) -> tan
Fall(ing) -> fall
Hiss(ing) -> hiss
Fizz(ed) -> fizz

(m=1 and *o) -> E
fail(ing) -> fail
Fil(ing) -> file

La regla para mapear a una sola letra provoca la eliminación de una letra doble. El –E es puesto en –AT, -BL y –IZ, de modo que los sufijos –ATE, -BLE y –IZE pueden ser reconocidos después. Esta E puede ser eliminada en el paso 4.

Paso 1c

(*v*) Y -> I	happy -> happi
	Sky -> sky

El paso 1 trata con plurales y pasados participios. Los pasos siguientes son mucho más sencillos.

Paso 2

(m>0) ATIONAL -> ATE	relational -> relate
(m>0) TIONAL -> TION	conditional ->condition
	Rational -> rational
(m>0)ENCI ->ENCE	valenci -> valence
(m>0) ANCI -> ANCE	hesitanci ->hesitance
(m>0) IZER -> IZE	digitizer -> digitize
(m>0) ABLI -> ABLE	conformabli -> conformable
(m>0) ALLI -> AL	radicalli -> radical
(m>0) ENTLI -> ENT	differentli -> different
(m>0) ELI -> E	vileli -> vile
(m>0) OUSLI ->OUS	analogousli -> analogous
(m>0) IZATION -> IZE	vietnamization -> vietnamize
(m>0) ATION ->ATE	predication -> predicate
(m>0) ATOR -> ATE	operator -> operate
(m>0) ALISM ->AL	feudalism -> feudal
(m>0) IVENESS -> IVE	dicisiveness -> decisive
(m>0) FULNESS -> FUL	hopefulness - > hopeful
(m>0) OUSNESS -> OUS	callousness -> callous
(m>0) ALITI -> AL	formaliti -> formal
(m>0) IVITI -> IVE	sensitivity -> sensitive
(m>0)BILITI -> BLE	sensibility -> sensible

La prueba para la cadena S1 se puede hacer rápido haciendo un programa que cambie la penúltima letra de la palabra que está siendo probada. Esto da una descomposición bastante justa de los posibles valores de la cadena S1. Esto se verá en el hecho de que

las cadenas S1 en el paso 2 son presentadas en el orden alfabético de la penúltima letra. Técnicas similares pueden ser aplicadas en los otros pasos.

Paso 3

(m>0) ICATE -> IC	triplícate -> triplic
(m>0) ATIVE ->	formative -> form
(m>0) ALIZE ->AL	formalize -> formal
(m>0) ICITI -> IC	electriciti -> electric
(m>0) ICAL -> IC	electrical -> electric
(m>0) FUL ->	hopeful -> hope
(m>0) NESS ->	goodness -> good

Paso 4

(m > 1) AL ->	revival -> reviv
(m > 1) ANCE ->	allowance -> allow
(m > 1) ENCE ->	inference -> infer
(m > 1) ER ->	airliner - > airlin
(m > 1) IC ->	gyroscopic -> gyroscop
(m > 1) ABLE ->	adjustable -> adjust
(m > 1) IBLE ->	defensible -> defens
(m > 1) ANT ->	irritant -> irrit
(m > 1) EMENT ->	replacement -> replac
(m > 1) ENT ->	dependent -> depend
(m > 1) and (*S or *T)ION ->	adoption -> adop
(m > 1) OU ->	homologou -> homolog
(m > 1)ISM ->	communism -> commun
(m > 1) ATE ->	activate -> activ
(m > 1) ITI ->	angulariti -> angular
(m > 1)OUS ->	homologous -> homolog
(m > 1) IVE ->	effective -> effect
(m > 1)IZE ->	bowdlerize -> bowdler

Los sufijos son ahora eliminados. El resto es un poco arreglado.

Paso 5a

(m > 1) E ->	probate -> probat
	Rate -> rate
(m = 1 and not *o)E ->	cease -> ceas

Paso 5b

(m > 1 and *d and *L) ->	una sola letra
	Controll -> control
	Roll -> roll

El algoritmo es cuidadoso de no eliminar un sufijo cuando la raíz es demasiado corta, la longitud de la raíz está dada por su medida m. No hay una base lingüística para este

enfoque. Solo fue observado que m podría ser bastante efectivo para ayudar a decidir si es o no prudente quitar un sufijo.

Lista A	Lista B
RELATE	DERIVATE
PROBATE	ACTIVATE
CONFLATE	DEMONSTRATE
PIRATE	NECESSITATE
PRELATE	RENOVATE

-ATE es eliminado de las palabras de la lista B, pero no de las palabras de lista A. Esto significa que los pares DERIVATE/DERIVE, ACTIVATE/ACTIVE, DEMOSTRATE/DEMOSTRABLE, NECESSITATE/NECESSITOUS, se combinarán a la vez. El hecho de que no se hace ningún esfuerzo para identificar los prefijos puede hacer que los resultados parezcan más bien incoherentes. Por lo tanto PRELATE no pierde el –ATE, pero ARCHPRELATE se vuelve ARCHPREL. En la práctica esto no es demasiado importante, porque la presencia del prefijo disminuye la probabilidad de una combinación errónea.

Los sufijos complejos son eliminados parte por parte en los diferentes pasos. De esta manera, GENERALIZATIONS es reducido a GENERALIZATION (paso 1), luego a GENERALIZE (paso 2), luego a GENERAL (paso 3), y luego a GENER (paso 4). OSCILLATORS es reducido a OSCILLATOR (paso 1), luego a OSCILLATE (paso 2), luego a OSCILL (paso 4), y luego a OSCIL (paso 5). En un vocabulario de 10.000 palabras, la reducción en el tamaño de la raíz fue distribuida entre los pasos como sigue:

La eliminación de sufijos de un vocabulario de 10.000 palabras
Número de palabras reducidas en el paso 1: 3597
Número de palabras reducidas en el paso 2: 766
Número de palabras reducidas en el paso 3: 327
Número de palabras reducidas en el paso 4: 2424
Número de palabras reducidas en el paso 5: 1373
Número de palabras no reducidas: 3650

El vocabulario resultante de las raíces contenía 6370 entradas distintas. De esta manera el proceso de eliminación redujo el tamaño del vocabulario aproximadamente en un tercio.

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz
Ph. D. Carlos Alberto Cobos Lozada

ANEXO C – ALGORITMO FPGROWTH

4 FPGROWTH

4.1 Introducción

Este algoritmo está basado en una representación de árbol de prefijos de una base de datos de transacciones llamada Frequent Pattern Tree (Árbol de Patrones Frecuentes) (Han, Pei et al. 2004). "La idea básica del algoritmo FP-Growth puede ser descrita como un esquema de eliminación recursiva: en un primer paso de pre-procesamiento se borran todos los ítems de las transacciones que no son frecuentes individualmente o no aparecen en el mínimo soporte de transacciones, luego se seleccionan todas las transacciones que contienen al menos un ítem frecuente, se realiza esto de manera recursiva hasta obtener una base de datos reducida. Al retorno, se remueven los ítems procesados de la base datos de transacciones en la memoria y se empieza otra vez, y así con el siguiente ítem frecuente" (Naranjo and Sierra 2009). "Los ítems en cada transacción son almacenados y luego se ordena descendentemente su frecuencia en la base de datos. Después de que se han borrado todos los ítems infrecuentes de la base de datos de transacciones, se pasa al árbol FP. Un árbol FP es básicamente de prefijos para las transacciones, esto es: cada camino representa el grupo de transacciones que comparten el mismo prefijo, cada nodo corresponde a un ítem. Todos los nodos que referencian al mismo ítem son referenciados juntos en una lista, de modo que todas las transacciones que contienen un ítem específico pueden encontrarse fácilmente y contarse al recorrer la lista. Esta lista puede ser accedida a través de la cabeza, lo cual también expone el número total de ocurrencias del ítem en la base de datos" (Naranjo and Sierra 2009).

Inserciones en la base de datos: este algoritmo no requiere de la generación de candidatos, por lo tanto, precisa de pocos accesos a la base de datos (Borgelt 2005).

Costo computacional: el algoritmo está basado en una representación de árbol de prefijos de una base de datos de transacciones, por lo tanto no necesita de la creación de un árbol de prefijos; sin embargo, la creación de dicho árbol no requiere de un costo computacional elevado (Han and Kamber 2006).

Tiempo de ejecución: este algoritmo busca patrones frecuentes con una corta búsqueda recursiva de prefijos, lo que en tiempo de ejecución es muy superior al del A priori, ya que no requiere de constantes accesos a la base de datos (Borgelt 2005).

Rendimiento: puede generar un árbol FP-Tree de una base de datos proyectada si el árbol inicial no se puede alojar completamente en la memoria principal, lo que le permite adecuarse a los recursos disponibles (Han and Kamber 2006).

De acuerdo a lo anterior, se decide implementar el algoritmo FPGrowth ya que tiene ventajas operacionales sobre los otros al no necesitar de la generación de ítems candidatos y ser computacionalmente más rápido. Entre las razones por la que se seleccionó este algoritmo tenemos que requiere de pocos accesos a la base. Este algoritmo está basado en una representación de árbol de prefijos de una base de datos de

transacciones; sin embargo, la creación de dicho árbol no requiere de un costo computacional elevado. El algoritmo busca patrones frecuentes con una corta búsqueda recursiva de prefijos, lo que en tiempo de ejecución es muy superior al A priori, ya que no requiere constantes accesos a la base de datos (Han and Kamber 2006).

4.2 Pasos del Algoritmo

El algoritmo FP-growth está compuesto de dos pasos importantes, que a su vez se dividen de otros pasos más específicos. En el primero gran paso se debe construir un Árbol de Patrones Frecuente o FP-tree y en el segundo paso se desarrolla un método basado en el FP-tree creado en el paso anterior, el método es llamado Crecimiento de Patrones Frecuentes (Frequent Patter-growth) ó FP-growth, el cual descubre los conjuntos completos de patrones frecuentes (Han, Pei et al. 2004).

A continuación se describen estos dos pasos importantes en el algoritmo FP-growth.

1. Diseño y construcción de un árbol de patrones frecuentes ó FP-tree: Un FP-tree es un árbol de patrones frecuentes, cuya estructura de árbol se define como sigue:

1. Consiste de una raíz etiquetada como “null”, un conjunto de sub-árboles de prefijos de ítems como el hijo de la raíz, y una tabla de cabecera de ítems frecuentes.

2. Cada nodo en el sub-árbol de prefijos de ítem consiste de tres campos: nombre de ítem, cuenta, y un enlace de nodo, donde el nombre del ítem registra el ítem que el nodo representa, la cuenta registra el número de transacciones representadas por la porción del camino que alcanza este nodo, y el enlace de nodo enlaza al siguiente nodo en el FP-tree que lleva el mismo nombre de ítem, o null si no hay uno.

3. Cada entrada en la tabla de cabecera de ítems frecuentes consiste de dos campos, (1) el nombre del ítem y (2) la cabeza del enlace de nodo, la cual apunta al primer nodo en el FP-tree que lleva el nombre del ítem.

Con base en esta definición, se tiene el siguiente algoritmo de construcción de FP-tree.

Algoritmo 1: Construcción del Árbol FP-tree

Entrada: Una base de datos de transacciones *DB* y un umbral de soporte mínimo ξ. *Salida*: Su árbol de patrones frecuentes, FP-tree. *Método*: El FP-tree es construido en los siguientes pasos.

1. Escanear la base de datos *DB* una vez. Reunir el conjunto de ítems frecuentes F y sus soportes. Ordenar F en orden descendente de soporte como L, la lista de ítems frecuentes.

2. Crear la raíz de un FP-tree, *T*, y etiquetarlo como “*null*”. Para cada transacción *Trans* en *DB* hacer lo siguiente.

Seleccionar y ordenar los ítems frecuentes en *Trans* de acuerdo al orden de *L*. Permitir que la lista ordenada de ítems frecuentes en *Trans* sea *[p | P]*, donde *p* es el primer elemento y *P* es la lista restante. Llamar a la función *insert_tree([p | P],T)*. La función *insert_tree([p | P],T)* se ejecuta como sigue. Si *T* tiene un hijo *N* tal que *N.item-name = p.item-name*, entonces incrementar la cuenta de *N* en 1; sino crear un nuevo nodo N, y permitir que su cuenta sea 1, su enlace de padre esté enlazado a *T*, y su enlace de nodo esté enlazado a los nodos con el mismo *ítem-name* por medio de la estructura de enlace de nodo. Si *P* no está vacío, llamar a *insert-tree(P,N)* recursivamente. Desde el proceso de construcción del FP-tree, se observa que solo se necesitan dos escaneos de la base de datos de transacciones, *DB*: el primero reúne el conjunto de ítems frecuentes, el segundo construye el FP-tree.

2. Descubrir los patrones frecuentes utilizando FP-tree

Después de construir el árbol FP-tree se prosigue con el siguiente algoritmo para la minería de patrones frecuentes.

Algoritmo 2: Método FP-growth

*Entrada***:** FP-tree construido con base en el Algoritmo 1, usando DB y un umbral de soporte mínimo ξ.
*Salida***:** El conjunto completo de patrones frecuentes.
Método: Llamar FP-growth(FP-tree, null).

```
Procedimiento FP-growth (Tree, α)
{
        (1) Si Tree contiene un solo camino P
        (2) Entonces para cada combinación (denotada como β) de los nodos en el
            camino P hacer
        (3)         generar el patrón β U α con soporte = soporte mínimo de nodos en β;
        (4) Sino para cada ai en la cabecera de Tree hacer {
        (5)         generar el patrón β = ai U α con soporte = ai.soporte;
        (6) construir la base de patrones condicionales de β y luego el FP-tree
            condicional de β o Treeβ;
        (7) si Treeβ ≠ Ø
        (8) entonces llamar FP-growth (Treeβ, β) }
}
```

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz
Ph. D. Carlos Alberto Cobos Lozada

ANEXO D – IMPLEMENTACION DE LUCENE.NET

5 LUCENE.NET

Debido a las necesidades específicas del presente proyecto, de la implementación de Lucenet.NET, solo se utilizó la funcionalidad correspondiente al proceso de indexación, en sus fases de análisis léxico, eliminación de palabras vacías y stemming; no se necesitó reutilizar más fases del indexado, ni la funcionalidad de búsqueda, puesto que esta última se implementa como parte del desarrollo del proyecto. Las clases que se utilizaron en la indexación fueron: *IndexWriter*, *Directory*, *Analyzer*, *Document*, *Field* e *IndexReader*. A continuación se presentan los pasos que se realizan en la fase de pre-procesamiento de documentos, en los cuales se reutiliza parte del código de Lucene.net:

1. Crear el índice: Se crea el índice, para lo cual se utilizan las clases *RAMDirectory* e *IndexWriter*. Se utiliza un directorio de tipo *RAMDirectory* porque no es necesario almacenar en disco los datos, ya que los temas de las consultas a la web por parte de los usuarios cambian mucho, por otra parte el tamaño de los datos no es tan elevado de modo que pueden ser cargados completamente en memoria RAM. Además, no era objetivo de esta investigación, realizar el indexado de documentos de la web, para ello se aprovecha el trabajo de buscadores tradicionales como Google.

2. Indexar los documentos: Se indexa el texto de cada documento en el índice que se creó:

2.1. Para indexar un documento primero se crea un objeto de la clase Analyzer, el objeto *StopAnalyzer*, el cual crea palabras o *tokens* de los caracteres que encuentra adyacentes en el flujo de caracteres que recibió como documento, además cada caracter que encuentra lo convierte en minúscula, por otra parte los números que encuentre no los tiene en cuenta, es decir no los incluye en ningún *token*. Luego el *StopAnalyzer* elimina las palabras que se encuentran en la lista de *stopwords* que Lucene ya trae especificada, aunque se puede modificar si se requiere, en nuestro caso modificamos esta lista la cual se puede ver en detalle en el Anexo A.

2.2. Después se crea un objeto *PorterStemFilter* que toma la raíz de cada palabra utilizando el algoritmo de stemming de Porter que viene implementado en Lucene, para ver en detalle el algoritmo remitirse al Anexo B.

2.3. Se crea un objeto *Field* que corresponderá al contenido del documento (resumen), este campo tendrá las propiedades *Store*, *Index*, *TermVector*, es decir el campo se almacenará, se indexará y mantendrá un vector de los términos del documento con sus frecuencias asociadas.

2.4. Se crea un objeto *Document* y a este se le adiciona el campo que se creó en el paso anterior.

2.5. Finalmente al objeto *IndexWriter* se le adiciona el documento, es decir el documento queda guardado en el índice con todas las características que se le asignó.

3. **Consultar en el índice**: Se utiliza el *IndexReader* para acceder al índice, para ello se debe especificar el *RAMDirectory* para ubicar el índice creado. Una vez abierto el índice se utiliza el objeto *TermFreqVector* para obtener los términos de los documentos.

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz
Ph. D. Carlos Alberto Cobos Lozada

ANEXO E – CASOS DE USO REALES

6 CASOS DE USO REALES

A continuación se presentan los casos de uso reales Iniciar Sesión (ver Tabla xxx) y Calificar Grupos y Documentos (ver Tabla xxx).

CASO DE USO REAL: INICIAR SESION	
Actores: Cliente.	
Propósito: El usuario inicia sesión en la aplicación.	
Resumen: El usuario inicia sesión en la aplicación, mediante el servicio de google.	
Tipo: Primario.	
CURSO NORMAL DE LOS EVENTOS	
Acción del actor	**Respuesta del sistema**
1. El usuario digita el correo [B], la contraseña [C] y da click en el botón Iniciar sesión [D].	2. El sistema lo redirige nuevamente al meta buscador y habilita la opción de búsqueda.
CURSO ALTERNO	
Acción del actor	**Respuesta del sistema**
2. El usuario no digita los campos obligatorios.	3. El sistema le informa que debe ingresar los campos.

Tabla 1. Caso de Uso Real Iniciar Sesión

CASO DE USO REAL: CALIFICAR GRUPOS Y DOCUMENTOS	
Actores: Cliente.	
Propósito: Calificar los grupos y los documentos.	
Resumen: El usuario a medida que selecciona un grupo y sus documentos puede realizar la calificación para cada caso.	
Tipo: Secundario.	
CURSO NORMAL DE LOS EVENTOS	
Acción del actor	**Respuesta del sistema**
1. El usuario da click en el botón de calificación, ubicado al lado de título del documento.	2. El sistema le muestra las opciones para calificar el documento.
3. El usuario califica el documento que ha seleccionado, según las opciones mostradas.	4. El sistema registra la calificación ingresada y cambia la imagen del botón de calificación.
5. El usuario califica el grupo actual, según las opciones mostradas.	6. El sistema registra la calificación ingresada.

Tabla 2. Caso de Uso Real Calificar Grupos y Documentos

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz
Ph. D. Carlos Alberto Cobos Lozada

ANEXO F – DIAGRAMA DE CLASES DEL SISTEMA

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz
Ph. D. Carlos Alberto Cobos Lozada

7 DIAGRAMA DE CLASES

En la Figura xxx se muestra de manera general el diagrama de clases del sistema y en la Tabla xxx se describe la funcionalidad de cada Clase.

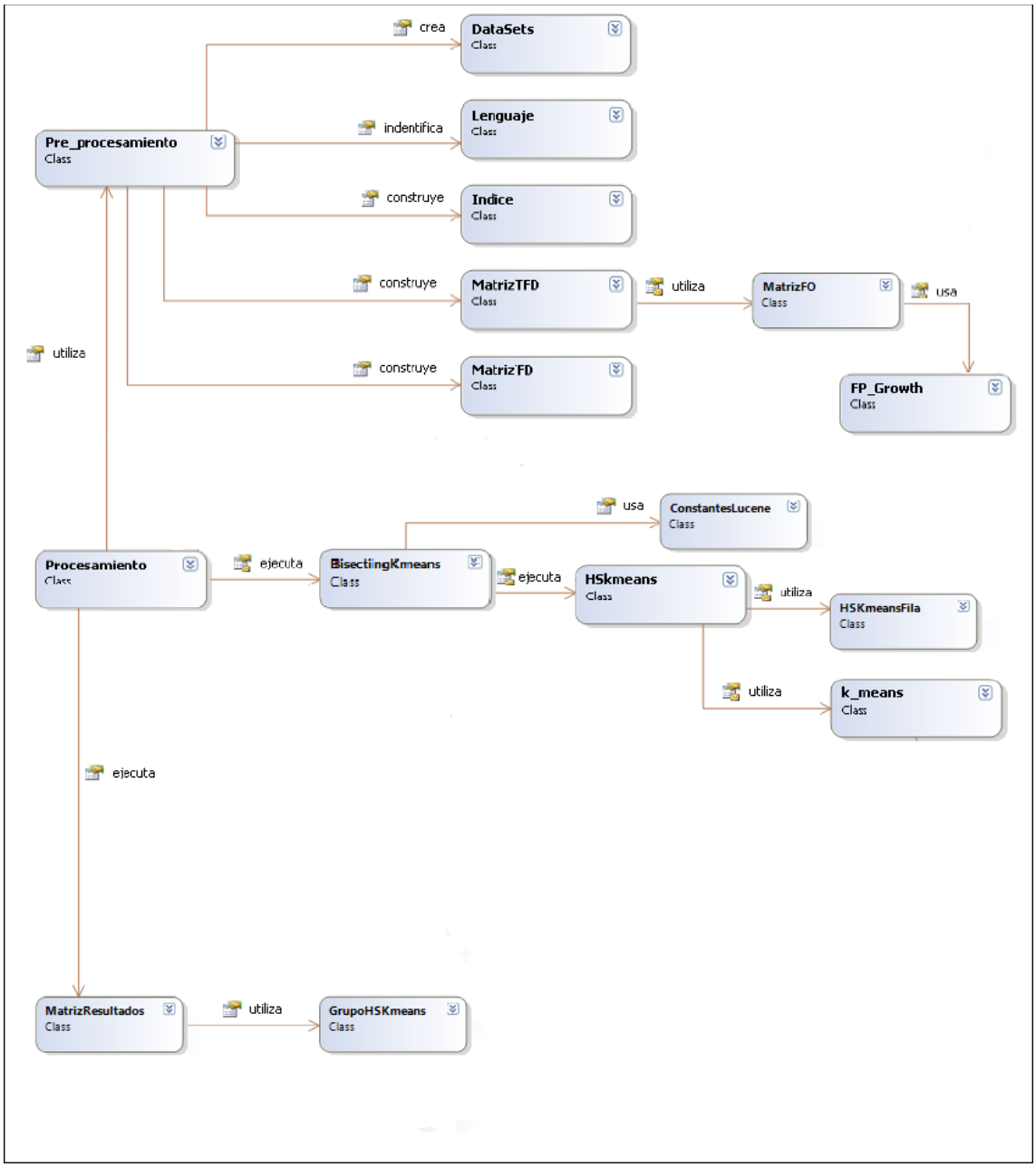

Figura 1. Diagrama General de Clases

Clase	Función
DataSets	Provee la funcionalidad para crear el Data Set con los documentos.
Índice	Provee la funcionalidad correspondiente a la indexación de los documentos tales como: análisis léxico, eliminación de palabras vacías y stemming. Igualmente provee la funcionalidad correspondiente a la obtención de las raíces de los documentos para el proceso de etiquetado.
Fp-Growth	Obtiene la lista de Itemsets frecuentes de los documentos.
Matriz TFD	Provee la funcionalidad para crear la matriz de términos frecuentes por documentos.
Matriz FD	Provee la funcionalidad para crear la matriz de Frases por documentos.
Pre-procesamiento	Provee toda la funcionalidad correspondiente al pre-procesamiento de los documentos.
ConstantesLucene	Mantiene los parámetros para el algoritmo.
K-means	Provee la funcionalidad del algoritmo k-means
MatrizResultados	Se encarga de mantener las mejores soluciones del algoritmo WDC-PTFBK.
Evaluacion	Provee la funcionalidad para el cálculo de las medidas de relevancia (Precisión, Exhaustividad y Medida F). esta clase se utiliza solamente para la evaluación con el dataset de Reuters.
BisectingKmeans	Provee la funcionalidad principal para la ejecución del algoritmo WDC-PTFBK.
MatrizResultado	Provee la funcionalidad correspondiente a la asignación de etiquetas, calificación de grupos y documentos, ordenamiento de grupos y documentos.
Procesamiento	Es la clase principal, que procesa la consulta del usuario para obtener los documentos, realizar el pre-

	procesamiento de los mismos, ejecutar el algoritmo, crear los resultados y procesar la calificación de grupos y documentos.
Frase	Identifica la frase de cada uno de los documentos.
GrupoHSKmeans	Tiene todos los datos para generar los clusters.
HSKmeans	Provee la funcionalidad correspondiente a la creación de una solución del algoritmo WDC-PTFBK.
HSKmeansFila	Provee la funcionalidad correspondiente a la Creación de Centroides Aleatorios y a la Creación de un Improviso.
MatrizFFD	Provee la funcionalidad para crear la matriz de frases frecuentes por documentos.
MatrizFO	Provee la funcionalidad para crear la matriz de frecuencias observadas.
MatrizKmeans	Almacena el resultado después del proceso de cluster.
Noticia	Almacena información importante de la noticia.
Termino	Almacena la frecuencia del término.

Tabla 3. Descripción de las clases.

8 CLASES DEL SISTEMA

A continuación se presentan en detalle las clases del sistema (ver Figura 2 a Figura 15).

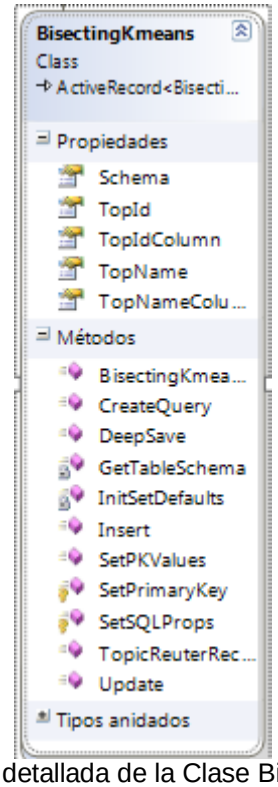

Figura 2. Vista detallada de la Clase BisectingKmeans

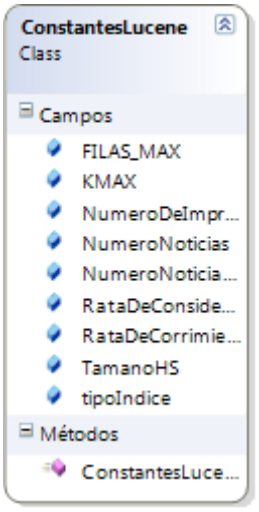

Figura 3. Vista detallada de la clase ConstantesLucene

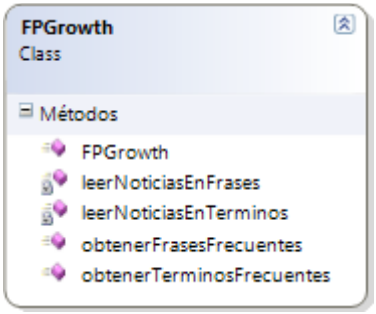

Figura 4. Vista detallada de la clase FPGrowth

Figura 5. Vista detallada de la clase GrupoHSKmeans

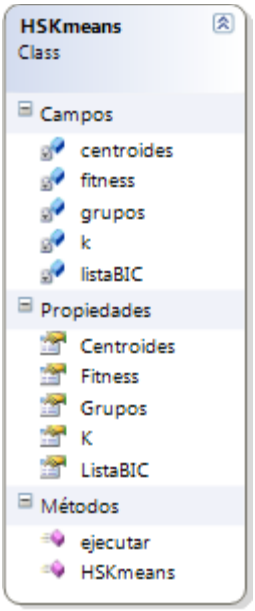

Figura 6. Vista detallada de la clase HSKmeans

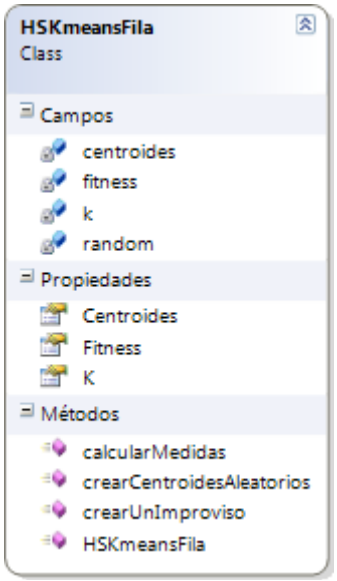

Figura 7. Vista detallada de la clase HSKmeansFila

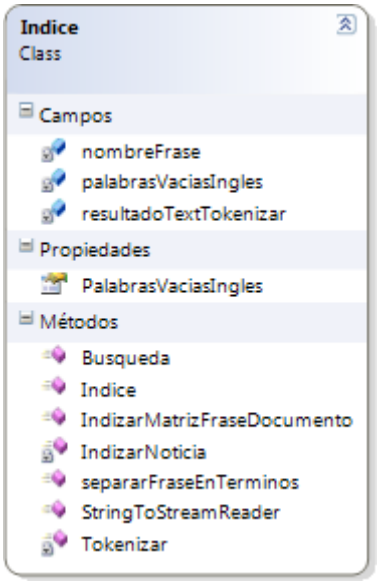

Figura 8. Vista detallada de la clase Indice

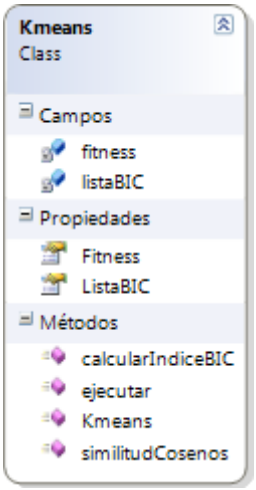

Figura 9. Vista detallada de la clase Kmeans

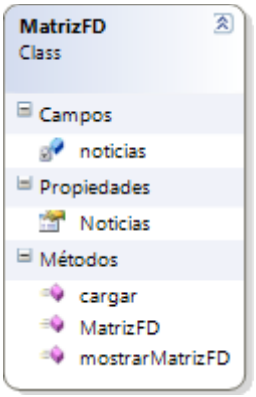

Figura 10. Vista detallada de la clase MatrizFD

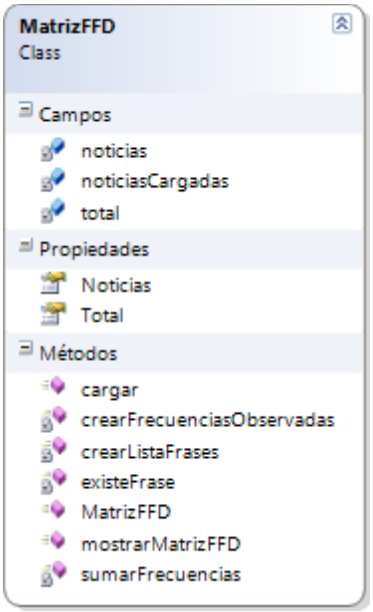

Figura 11. Vista detallada de la clase MatrizFFD

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz
Ph. D. Carlos Alberto Cobos Lozada

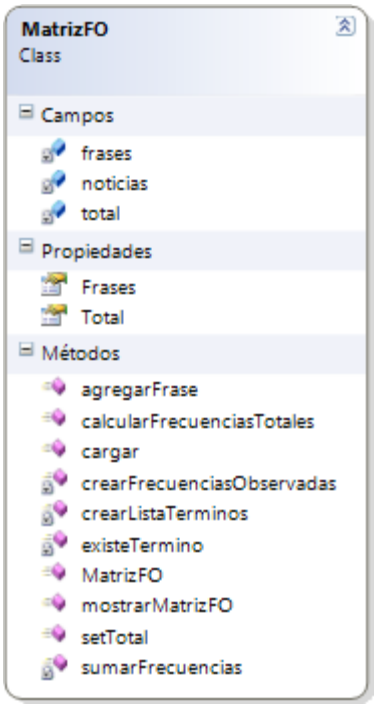

Figura 12. Vista detallada de la clase MatrizFO

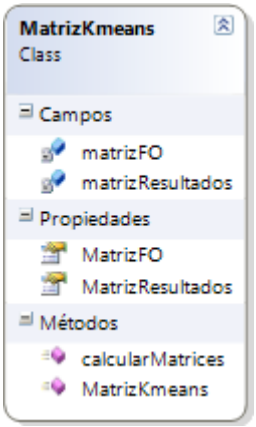

Figura 13. Vista detallada de la clase MatrizKmeans

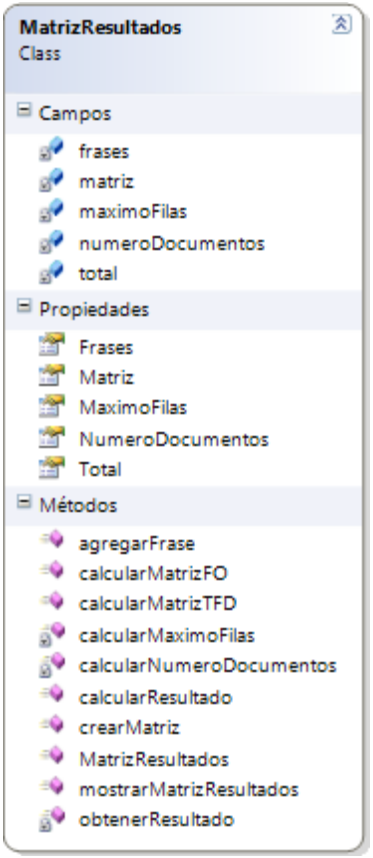

Figura 14. Vista detallada de la clase MatrizResultados

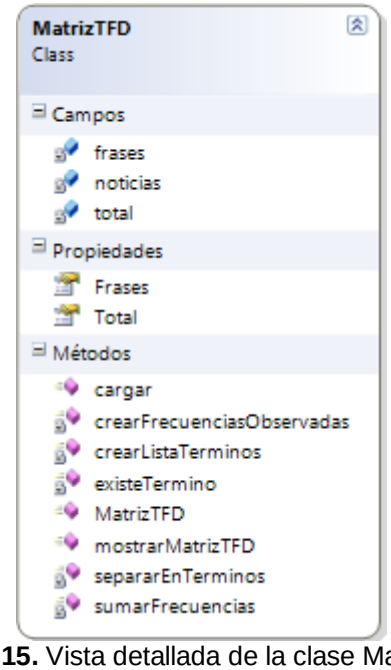

Figura 15. Vista detallada de la clase MatrizTFD

Daniel Andrés Pino Gómez
Pablo Alejandro Zúñiga Muñoz
Ph. D. Carlos Alberto Cobos Lozada

ANEXO G – SURVEY

UN SURVEY DE REGLAS DE ASOCIACIÓN EN MINERÍA DE DATOS

Carlos Alberto Cobos Lozada
ccobos@unicauca.edu.co
Daniel Andrés Pino Gómez
dapino@unicauca.edu.co
Pablo Alejandro Zúñiga Muñoz
pabloz@unicauca.edu.co

REFERENCIAS

Borgelt, C. (2005). "An implementation of the FP-Growth Algorithm."

Han, J. and M. Kamber (2006). Data Mining: Concepts and Techniques, Morgan Kaufman Publishers.

Han, J., J. Pei, et al. (2004). "Mining Frequent Patterns without Candidate Generation A Frequent Pattern Tree Approach."

Naranjo, R. and L. Sierra (2009). "Herramienta software para el análisis de canasta de mercado sin selección de candidatos."