RNPS No. 2143
ISSN 2072-6260
Versión Digital

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

REPORTE TÉCNICO
Minería de Datos

SERIE GRIS

Segmentación lineal de texto por tópicos

Raúl Abella Pérez y José E. Medina Pagola

RT_025 **mayo 2014**

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

RNPS No. 2143
ISSN 2072-6260
Versión Digital

SERIE GRIS

REPORTE TÉCNICO
Minería de Datos

Segmentación lineal de texto por tópicos

Raúl Abella Pérez y José E. Medina Pagola

RT_025 **mayo 2014**

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

Segmentación lineal de texto por tópicos

Raúl Abella Pérez y José E. Medina Pagola

Equipo de Investigaciones de Minería de Datos, Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV), La Habana, Cuba
{rabella, jmedina}@cenatav.co.cu

RT_025, Serie Gris, CENATAV
Aceptado: 20 de enero de 2014

Resumen. La segmentación automática lineal de textos, con el fin de detectar los límites de tópicos, es una tarea muy difícil y útil en muchos sistemas de procesamiento de textos. Muchos métodos tratan de resolver este problema obteniendo resultados razonables, pero estos presentan algunas dificultades, como son: incorrecta interrupción de los segmentos, no incluyendo oraciones o párrafos que pertenecen al segmento, generando así segmentos con información incompleta. Cuando esta situación ocurre, se obtienen segmentos espurios. Un paso importante en la segmentación de texto por tópicos es la selección del modelo de representación de texto. En este trabajo se evalúan diferentes modelos de representación de texto que se han utilizado en la segmentación de texto por tópicos. Además, se analizan algunos de los principales métodos de segmentación de texto por tópicos, que utilizan dichos modelos.

Palabras clave: segmentación lineal por tópicos, cohesión léxica, LSA, PLSA, LDA.

Abstract. Automatic linear text segmentation, in order to detect the best topic boundaries, is a task very difficult and useful in many text processing systems. Some methods have tried to solve this problem with reasonable results, but they present some difficulties as, for instance, wrong interruptions of segments, leaving out sentences or paragraphs which belong to the segments, and generating segments with incomplete information. When these situations happen, spurious segments are obtained. An important step in text segmentation by topic is the selection of text representation model. In this paper we analyze different text representation models that have been used in the text segmentation by topic. In addition, we discuss some of the text segmentation main by topic that uses these models.

Keywords: linear segmentation by topics, lexical cohesion, LSA, PLSA, LDA.

1 Introducción

El surgimiento y desarrollo de las computadoras ha tenido un gran impacto en la vida del hombre, interactuando prácticamente en todas las esferas: medicina, educación, en la comunicación, etc. Las computadoras son capaces de almacenar grandes volúmenes de información, permitiendo su conservación con el paso del tiempo. Con el objetivo de facilitar el manejo y comprensión de estos volúmenes de información surgen las herramientas para el procesamiento de textos; por ejemplo, la

recuperación de información (IR, por sus siglas en inglés), la generación automática de resúmenes, la detección y seguimiento de tópicos, entre otras.

La segmentación de textos juega un papel importante en la eficacia y la eficiencia de estas herramientas. Por ejemplo, en la recuperación de información, los métodos de segmentación por tópicos son útiles, ya que se necesita devolver todos los segmentos o pasajes más relacionados con la consulta realizada por el usuario en lugar del documento completo. En las herramientas de generación automáticas de resúmenes, también es importante la segmentación de textos por tópicos, ya que si se tiene conocimiento de todos los sub-tópicos que forman un documento, estos se pueden utilizar para seleccionar las ideas principales que conformarían el resumen del documento [1-5].

Algunos autores plantean que la segmentación de texto por párrafos también es de gran utilidad en las herramientas de confección automática de resúmenes de múltiples documentos. Por ejemplo, en la generación de resúmenes de múltiples documentos de la web [6]; ya que, en la salida los textos no retienen necesariamente la estructura correcta de párrafos y pueden que requieran de un post-procesado.

La segmentación de textos puede ser definida como la tarea de dividir un documento en unidades sintácticas (párrafos, oraciones, etc.) o en bloques semánticos, usualmente en tópicos [5]. La dificultad de la segmentación depende principalmente de las características del documento que se quiere segmentar (ejemplo: textos científico-técnico, cuentos, etc.) y de lo que se quiere obtener como resultado de la segmentación (ejemplo: tópicos, párrafos, oraciones, etc.). Existen varios acercamientos para dar solución a este problema como: la segmentación lineal, donde el documento de entrada es dividido en una secuencia lineal de segmentos adyacentes. Otro enfoque es la segmentación jerárquica; la salida de estos algoritmos es parecida a la típica estructura de un documento que consta de capítulos y múltiples niveles de sub-capítulos hasta el nivel de párrafo. El presente trabajo se enfoca en la segmentación de textos por tópicos, específicamente en los métodos que hacen una segmentación lineal.

En la actualidad existen varios métodos de segmentación lineal de textos por tópicos [1, 2, 5, 7]. La mayoría de estos enfoques explotan la información léxica. En sentido general, la principal dificultad de estos enfoques es que las unidades textuales (oraciones, párrafos o bloques) son representadas por el Modelo Espacio Vectorial. Este modelo se basa en una comparación estricta de los términos (tras su lematización o la extracción de raíces léxicas), por lo que la eficacia se ve afectada por palabras distintas que describen el mismo concepto (sinonimia) y por palabras con distintos significados (polisemia). Para superar estos problemas algunos métodos utilizan modelos o técnicas que reflejan en mayor medida el contenido semántico de los documentos. Algunos de estos modelos son: Latent Semantic Analysis (LSA), Probabilistic Latent Semantic Analysis (PLSA) y Latent Dirichlet Allocation (LDA). Estos modelos a diferencia del Modelo de Espacio Vectorial hacen una reducción de la dimensión de los vectores, pero difieren en cómo hacen dicha reducción.

Independientemente del modelo que utilicen los métodos de segmentación, estos se pueden clasificar en dos grandes grupos. Por una parte se encuentran los métodos de segmentación “golosos”, los cuales de una sola pasada identifican los límites de segmento, tales como: TextTiling, ClustSeg, C99 y el propuesto por Thorsten Brants y autores. Por otro lado, están los métodos que utilizan programación dinámica, los cuales hacen varias segmentaciones con el objetivo de encontrar la “segmentación óptima”.

En sentido general los métodos de segmentación de texto por tópicos cuentan con las siguientes etapas: Pre-procesamiento, Representación del documento y por ultimo Identificación de los límites. En el presente trabajo se hace un análisis de como diferentes métodos de la literatura hacen cada una de estas etapas, además se hace una propuesta de una taxonomía de los métodos de segmentación.

En el presente trabajo se realiza un análisis crítico de los modelos antes mencionados, los cuales han sido utilizados en la tarea de segmentación de textos por tópico. Además, se analizan algunos de los principales métodos de segmentación de texto por tópicos que utilizan dichos modelos, señalando en cada caso sus ventajas y deficiencias. Este está estructurado de la siguiente forma: En la Sección 2 se describen algunos de los pasos que se realizan en el pre-procesamiento de los textos (aplicable a todos los métodos). En la Sección 3 se analizan los modelos de representación de textos que se han utilizado en la segmentación de texto. En la sección 4 se hace un análisis de las diferentes estrategias utilizadas

por los métodos de segmentación para identificar los límites de segmento. En la sección 5 se propone una taxonomía de los métodos de segmentación y además estos son analizados. Por último, se describen las experimentaciones realizadas y las conclusiones del reporte.

2 Pre-procesamiento

El pre-procesamiento del texto es una etapa requerida antes de comenzar cualquier tarea de procesamiento textual. La segmentación de texto también tiene un papel importante como parte del pre-procesamiento en algunas tareas, ya que el texto sufre una serie de transformaciones necesarias que facilitan la extracción de información que ayuda a la detección de unidades textuales relacionadas con un mismo tópico. Las transformaciones que ocurren en esta etapa varían en dependencia de los intereses de los autores y tienen como objetivo modificar el documento de forma tal que solo esté compuesto por un conjunto reducido de términos, los cuales representen el documento.

Para la mayoría de las tareas de procesamiento de texto lo que suele ser importante es el contenido del mismo y no su formato. Por tal motivo es necesario hacer la conversión a un formato de documento más ligero (texto plano), el cual ayuda a aumentar el rendimiento de los sistemas de pre-procesamiento de textos. El texto plano, texto llano, o texto simple son sólo caracteres, sólo texto sin formatear; es decir, sin códigos de tipos de letras, negritas, cursivas, formatos de párrafos, etc.

Entre las operaciones más comunes que se realizan en el pre-procesamiento de los documentos se encuentran:

- Eliminación de signos de puntuación, espaciados, reducción de mayúsculas, reconocimiento del formato del documento (eliminación de etiquetas si es una página HTML), así como el reconocimiento de palabras (tokenización).
- Eliminación de las palabras vacías o *stop-words*. Las *stop-words* son las palabras que se consideran carentes de utilidad para determinar semejanza entre las unidades textuales debido a su excesiva frecuencia en el documento, anulando su capacidad discriminante, generalmente son los artículos, preposiciones, conjunciones, etc. La eliminación de estas palabras varía en dependencia del autor; una de las formas más utilizada es la utilización de una lista de términos vacíos construida previamente. Esta lista está formada generalmente por preposiciones, conjunciones, artículos, pronombres y cualquier palabra que se requiera incluir por su elevada aparición.
- Extracción de raíces o lemas. Esto se hace con el objetivo de obtener en el mínimo número de caracteres posibles, el máximo de información del término. La raíz léxica o lexema es la unidad léxica primaria de una palabra, que lleva los aspectos más significativos del contenido semántico y que no se puede reducir en componentes más pequeños. Mientras que el lema relaciona a un conjunto de palabras con la misma raíz léxica, y que pertenecen a la misma categoría gramatical (verbo, adjetivo, etc.). La lematización tiene como objetivo normalizar los términos pertenecientes a una misma familia y por tanto próximos en significado, reduciéndolos a una forma común o lema, que no coincide necesariamente con la raíz. Este proceso comprende la eliminación de los plurales, de ciertos prefijos y sufijos, de las conjugaciones verbales y su reducción al infinitivo, etc. Una de las herramientas más utilizada para este proceso es el Treetager.

3 Representación del documento

Realizar la segmentación de textos en tópico, de forma automática como lo hace una persona es muy difícil, ya que para esto hay que tener en cuenta tanto la información sintáctica como la semántica. Por

tales motivos, es de gran importancia la correcta selección de un modelo de representación de texto que tenga en cuenta ambos aspectos.

El modelo espacio vectorial es uno de los más utilizados en la segmentación de texto por tópicos, principalmente por su eficiencia para determinar similitud entre unidades textuales [1-5, 7-9]. Este modelo se ha utilizado de diferentes formas en la literatura; por ejemplo, Hearst en 1994 propuso un método de segmentación, el cual utiliza este modelo para representar bloques de oraciones adyacentes. A diferencia de Hearst, Choi en el 2000 propone un método donde utiliza este modelo para representar las oraciones, en lugar de grupos de oraciones. Por otra parte, a diferencias de estos trabajos, en el 2004 Hernández y Medina proponen un método donde utilizan este modelo para representar párrafos adyacentes, similar a lo propuesto por Abella y Medina en el 2010. En sentido general la principal dificultad de este modelo es que se basa en una comparación estricta de los términos, por lo que la eficacia se ve afectada por palabras distintas que describen el mismo concepto (sinonimia) y por palabras con distintos significados (polisemia).

Con el objetivo de superar los problemas generados por la sinonimia y la polisemia; Choi propone en el 2001 un método de segmentación de texto por tópico utilizando el modelo LSA (*Latent Semantic Analysis*) [10], donde considera las oraciones como unidades mínimas, similar a lo propuesto por él en el 2000. En la literatura existen otros modelos que se han utilizado en la segmentación de texto, los cuales tratan de reflejar en mayor medida el contenido semántico de los documentos, tales como: *Probabilistic Latent Semantic Analysis* (PLSA) y *Latent Dirichlet Allocation* (LDA) [11-13]. En el trabajo propuesto por Brants y autores en el 2002 utilizan el modelo PLSA para representar bloques de oraciones adyacentes, al igual que en el trabajo propuesto por de Mimi Lu y autores en el 2011. Similar a estos autores, Hemant Misra propone un método en el 2009 donde representa bloques de oraciones, pero utilizando el modelo LDA. En estos modelos a diferencia del Modelo Espacio Vectorial se hace una reducción de dimensión de los vectores y además necesitan de una etapa de entrenamiento de datos.

A continuación serán descritos con más detalles los modelos antes mencionados.

3.1 Modelo espacio vectorial (VSM)

Como se mencionó anteriormente el Modelo de Espacio Vectorial (VSM) es uno de los modelos más utilizados por los algoritmos que procesan documentos. Mediante el VSM las unidades textuales se transforman en vectores dentro de un espacio multidimensional, donde las componentes son los términos diferentes resultantes del pre-procesamiento. Considerando como unidad textual el párrafo, un documento formado por n párrafos, $P = \{p_1, p_2 \dots p_n\}$ y por un conjunto de m términos únicos, $T = \{t_1, t_2 \dots t_m\}$, el párrafo i puede modelarse como un vector de la siguiente forma:

$$\overrightarrow{p_i} = \big(w(t_1, p_i), \dots, w(t_m, p_i)\big). \quad (1)$$

Donde $w(t_j, p_i)$ es el peso del término t_j en el párrafo p_i. El peso de cada término representa la importancia de este para representar el párrafo o cualquier otra unidad textual o documento. Existen diferentes formas de calcular el peso, entre las más utilizadas se encuentran:

- **Ponderado booleano:** Cada término t_j es asociado con un valor de 0 o 1. Toma el valor 1 si el término ocurre en el párrafo (u otra unidad textual o documento) y 0 en caso contrario.
- **Ponderado por frecuencia de término:** Cada término t_j es asociado con la frecuencia de aparición en el párrafo p_i (u otra unidad textual o documento), absoluta o normalizada. Generalmente esta frecuencia de aparición es normalizada con el objetivo de que este peso asociado no dependa de la frecuencia relativa con otros términos y para evitar favorecer a párrafos (u otra unidad textual o documento) más largos. Existen diferentes tipos de normalización reportados en la literatura. Una muy utilizada es la Normalización por la

longitud del documento, que consiste en dividir la frecuencia de cada término por la cantidad de términos presente en el documento.

- **Ponderado *tf-idf*:** Cada término t_j es asociado con un valor en proporción al número de ocurrencias en el párrafo p_i (u otra unidad textual o documento), y en proporción inversa al número de párrafos (u otra unidad textual o documento) para los cuales ocurre dicho término al menos una vez. En otras palabras tomando a N como el número de párrafos de un documento y n_i el número de párrafos donde aparece el término, el peso del término t_j en el párrafo p_i se calcularía de la siguiente forma:

$$w_{ij} = f_{ij} * log\, N/n_i\ . \tag{2}$$

Donde f_{ij} es la frecuencia en el párrafo p_i.

Una vez representados dos párrafos se puede determinar su cohesión léxica, utilizando una medida de similitud. Existen diferentes medidas como son: medida del coseno, Dice y Jaccard. En el presente trabajo abordaremos sobre la medida del coseno por ser la más utilizada en la segmentación de texto.

Para hallar la similitud entre dos párrafos p_i, p_j utilizando la medida del coseno quedaría de la siguiente forma:

$$sim(p_i, p_j) = cos(p_i, p_j) = \ (\overrightarrow{p_i} \cdot \overrightarrow{p_j})/\|\overrightarrow{p_i}\| * \|\overrightarrow{p_j}\| = \frac{\sum w_{ir} * w_{jr}}{\sqrt{\sum w_{ir}^2 * \sum w_{jr}^2}}\ . \tag{3}$$

Donde w_{ir}, w_{jr} es el componente r del vector p_i, p_j respectivamente.

3.2 LSA

Latent Semantic Analysis (LSA) propuesto en 1988 es una extensión del Modelo de Espacio Vectorial [14]. Este modelo surge con el objetivo de superar las dificultades semánticas, generadas por la sinonimia y la polisemia.

Este método asume que existe cierta asociación entre los términos utilizados en los documentos de una colección. Además considera que esta asociación puede ser estimada haciendo uso de técnicas estadísticas. En particular, hace uso de la descomposición de valores singulares (*Singular Value Descomposition*); con ella se pretende medir la similitud entre diferentes términos de un vocabulario. La SVD se puede considerar también como un método de reducción de rasgos, ya que permite reducir la dimensión de las representaciones.

El primer paso consiste en representar el texto como una matriz *M* en la que cada fila representa una palabra única (*p*), y cada columna representa una oración, párrafo o documento (*d*). Cada celda contiene la frecuencia con la cual la palabra de la fila aparece en el pasaje de texto denotado por su columna (*p*, *d*). Este valor puede ser transformado utilizando una función de peso que exprese la importancia del término en el pasaje o documento.

El segundo paso es aplicar a la matriz resultante el método SVD. Toda matriz $M \in \mathbb{R}^{m \times n}$ puede ser escrita como el producto de otras tres matrices:

$$M = U\Sigma V^T\ , \tag{4}$$

donde $\Sigma \in \mathbb{R}^{mxn}$ es una matriz diagonal de valores singulares, que una vez ordenados descendentemente $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0$ siendo $k = min\{m, n\}$ y las matrices *U* y *V* son matrices cuadradas formadas por vectores singulares, además estas matrices son ortogonales. Los valores singulares son únicos, y el número de valores singulares distintos de cero es el rango *r* de la matriz *M*, siendo $r \leq k$.

Una vez que se dispone de una descomposición en valores singulares de la matriz *M*, es posible aproximarla por otra matriz M_p de rango $p \leq r$. Para ello se ordena la matriz diagonal y las matrices *U* y *V* descendentemente y se toman los *p* mayores valores singulares:

$$M_p = U_p \Sigma_p V_p^T \,, \tag{5}$$

donde U_p y V_p son matrices formadas por las *p* primeras columnas asociadas con los valores singulares seleccionados de las matrices *U* y *V*. Se asegura que M_p es una de las matrices de rango *p* o menor que mejor se aproxima a la matriz original *M*.

Una vez generado el espacio reducido se pueden realizar diferentes cálculos de similitud semántica con los vectores de dicho espacio. Por ejemplo, para medir cuan similar son dos documentos (párrafos u oraciones) *j* y *q* basta con comparar los vectores $\sum_p d_j$ y $\sum_p d_q$, donde d_j y d_q son los vectores de dichos documentos en el espacio reducido, es decir, en la matriz V_p. Para comparar dichos vectores se utiliza una métrica, normalmente el coseno. También se puede medir cuan similar es un documento *h (*párrafo u oración), que no se encuentra en el corpus de entrenamiento con otros documentos que se encuentran dentro del corpus de entrenamiento. Para esto, primero se lleva dicho párrafo al espacio reducido de la siguiente manera:

$$h' = \sum_p^{-1} U_p^T h \,, \tag{6}$$

y luego dicho vector se compara con el documento, o los documentos, de igual manera que se explicó anteriormente.

3.3 PLSA

Probabilistic Latent Semantic Analysis (PLSA) fue propuesto en 1999 por Hofmann como una versión de LSA [11]. Esta técnica se ha utilizado en aplicaciones en la recuperación de información, clasificación de imágenes, segmentación de textos, entre otros. PLSA se basa en un "Modelo de Aspecto", el cual asocia una variable latente no observada $z \in Z = \{z_1, \dots, z_K\}$ con cada observación, es decir, la ocurrencia de una palabra $w \in W = \{w_1, \dots, w_M\}$ en un documento en particular $d \in D = \{d_1, \dots, D_M\}$. El objetivo de este modelo es encontrar el mejor conjunto de variables latentes Z que puedan explicar de mejor manera los datos observados (es decir, las palabras en los documentos) asumiendo el modelo actual general de los datos. El modelo generativo es definido de la siguiente manera:

- Se selecciona un documento d con una probabilidad *P(d).*
- Se elige una clase latente *z* con probabilidad *P(z|d).*
- Se genera una palabra *w* con probabilidad *P(w|z).*

Como resultado se obtiene un par observado *(d,w)* mientras que la clase latente *z* es descartada. Trasladando este proceso a un modelo de probabilidad conjunta se obtiene la siguiente expresión:

$$P(d,w) = P(d)P(w/d) \,, \tag{7}$$

$$P(w/d) = \sum_{z \in Z} P(w/z)P(z/d) \,. \tag{8}$$

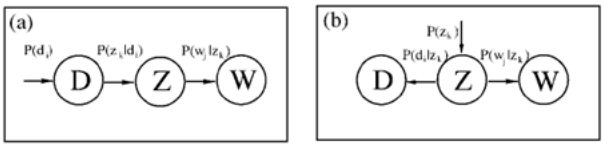

Fig. 1. Representación gráfica del modelo en la parametrización asimétrica (a) y simétrica (b).

El modelo de aspecto está basado en dos suposiciones de independencia, la primera de que el par *(d,w)* es generado en forma independiente y la segunda es la suposición de independencia condicional de la variable latente *z*; además las palabras *w* se generan independientemente del documento específico *d*. La representación correspondiente al modelo gráfico se representa en la Figura 1 (a). Además, este modelo puede ser equivalentemente parametrizado simétricamente por (Figura 1 (b)):

$$P(d,w) = \sum_{z \in Z} P(z)P(d/z)P(w/z)\,. \tag{9}$$

Los parámetros del modelo son estimados utilizando el algoritmo de Expectation - Maximization (EM). El algoritmo EM es un método iterativo que permite encontrar un máximo local de los parámetros del modelo. EM asume que los parámetros de un modelo son fijos y pueden estimarse de las frecuencias de los valores observados en experimentos que se repiten varias veces. El algoritmo EM alterna dos pasos:

- **E-Paso**. Paso de expectación, donde las probabilidades a posteriori son calculadas para las variables latentes z's basado en la estimación de los parámetros.
- **M-Paso**. Paso de maximización donde los valores de los parámetros son actualizados.

A continuación se presentan los pasos del algoritmo EM:

- Se inicializan los parámetros $P(w|z)$, $P(d|z)$ y $P(z)$ con números aleatorios en el intervalo [0,1].

1. E-Paso

$$P(z|d,w) = \frac{P(z)P(d|z)P(w|z)}{\sum_{z'} P(z')P(d|z')P(w|z')}\,. \tag{10}$$

2. M-Paso

$$P(w|z) = \frac{\sum_d n(d,w)\,P(z|d,w)}{\sum_{d,w'} n(d,w')P(z|d,w')}\,, \tag{11}$$

$$P(d|z) = \frac{\sum_w n(d,w)\,P(z|d,w)}{\sum_{d',w} n(d',w)P(z|d',w)}\,, \tag{12}$$

$$P(z) = \frac{\sum_{d,w} n(d,w)\,P(z|d,w)}{\sum_{d,w} n(d,w)}\,. \tag{13}$$

3. Si la prueba de convergencia no se cumple, actualizar los parámetros y regresar al paso 1.

$$P(w|z)^{old} \leftarrow P(w|z)^{new}\ P(d|z)^{old} \leftarrow P(d|z)^{new} \quad P(z)^{old} \leftarrow P(z)^{new}\ .$$

3.4 LDA

LDA es un modelo generativo de tópico no supervisado propuesto por Blei y Col en el 2002 [15]. Este modelo se ha utilizado en tareas de modelado de texto, clasificación de texto y el filtrado colaborativo, entre otras tareas.

LDA relaciona a las palabras y documentos a través de los tópicos latentes, basado en el supuesto de "saco de palabras"; es decir, intercambiabilidad de las palabras de un documento y de los documentos en un corpus. Los documentos θ no están directamente vinculados a las palabras *w*; más bien, esta relación se rige por las variables latentes adicionales, *z*, introducida para representar la responsabilidad de un tópico en particular en el uso de esa palabra en el documento, es decir, el(los) tópico(s) en que el documento se centra. Mediante la introducción de las distribuciones Dirichlet previas, α y β, sobre las distribuciones de documentos y tópicos, respectivamente, el modelo generativo de LDA es completo y es capaz de procesar documentos no vistos. A continuación se muestra el proceso generativo de LDA:

1. Seleccionar *K* multinomios (multinomials) ϕ_k de la distribución Dirichlet previa β, uno para cada tópico k.
2. Seleccionar *D* multinomios (multinomials) θ_d de la distribución Dirichlet previa α, uno para cada documento *d*.
3. Para cada documento d en el corpus, y para cada palabra w_{di} en el documento:
    a. Seleccionar un tópico z_i de multinomial θ_d; $\big(p(z_i|\alpha)\big)$;
    b. Seleccionar una palabra w_i de multinomial ϕ_z; $\big(p(w_i|z_i,\beta)\big)$.

Invirtiendo el proceso generativo, es decir, ajustando el modelo de variable oculta a los datos observados (palabras en un documento), permite inferir las variables latentes. La estructura oculta de tópicos en LDA se describe por la distribución posterior de las variables ocultas dados los documentos *D:*

$$p(\Theta, z, \Phi|w, \alpha, \beta) = \frac{p(w, \Theta, z, \Phi|\alpha, \beta)}{\int_{\phi_{1:K}} \int_{\theta_{1:D}} p(w|\alpha, \beta)}. \tag{14}$$

En LDA, la exploración de los datos y la extracción de los tópicos corresponden a calcular las expectativas posteriores. Estas son; la probabilidad de los tópicos sobre los términos $\big(E(\Phi|w)\big)$, las proporciones de los documentos sobre los tópicos $\big(E(\Phi|w)\big)$, y las asignaciones de tópicos de las palabras $\big(E(z|w)\big)$. La inferencia de las distribuciones posteriores de la ecuación (14) se hacen utilizando aproximaciones. Para ello existen diferentes métodos, tales como: *Expectation-Maximization*, *Expectation Propagation* o el muestreo de Gibbs.

El muestreo de Gibbs es una técnica de aproximación iterativa, la cual es una forma especial de la cadena de Markov Monte Carlo (MCMC), utilizada para la estimación de los parámetros. El muestreo de Gibbs es capaz de simular una distribución de probabilidad de alta dimensión $p(x)$, por muestreo iterativo de una dimensión x_i a la vez, condicionado a los valores de todas las otras dimensiones; se denota generalmente como $x_{\neg i}$.

En el muestreo de Gibbs, los parámetros no están explícitamente estimados. En cambio, la distribución posterior sobre las asignaciones de las palabras a los tópicos, $p(z|w)$, se aproxima por medio del algoritmo de Monte Carlo, ver Heinrich (2005) para una conocimiento detallado del

algoritmo. El muestreo de Gibbs itera sobre cada palabra en la colección de texto en un orden aleatorio y estima la probabilidad de asignar la palabra actual a cada tópico $(p(z_i = j))$, condicionado a la asignación de tópicos a todas las otras palabras $z_{\neg i}$.

$$P(z_i = j | z_{\neg i}, w, \alpha, \beta) \propto \frac{C^{KW}_{w_{\neg i},j} + \beta_{w_{di},j}}{\sum_{v=1}^{W}\left(C^{KW}_{v,j} + \beta_{v,j}\right)} \times \frac{C^{KD}_{d_{\neg i},j} + \alpha_{d,j}}{\sum_{k=1}^{K}\left(C^{KD}_{d,k} + \alpha_{d,k}\right)}, \quad (15)$$

donde $C^{KW}_{w_{\neg i},j}$ es el número de veces que la palabra *w* es asignada al tópico *j*, sin incluir la instancia de *token* actual *i*; y $C^{KD}_{d_{\neg i},j}$ es el número de veces que el tópico *j* es asignado a alguna palabra en el documento *d*, sin incluir la instancia actual *i*. De esta distribución, $p(z_i | z_{\neg i}, w)$, un tópico se muestrea y se almacena como la asignación de un nuevo tópico para esta palabra. Después de un número suficiente de iteraciones de muestreo, la aproximación posterior se puede utilizar para obtener las estimaciones de φ y θ, examinando las asignaciones de las palabras a los tópicos y las ocurrencias de los tópicos en los documentos.

La estimación directa de las asignaciones de tópico *z* para cada palabra es importante para obtener su relación con los parámetros requeridos Θ y Φ. Esto se logra mediante el muestreo de nuevas observaciones basado en el estado actual de la cadena de Markov. Por lo tanto, la estimación de Θ' y Φ' de las distribuciones, palabra - tópico y tópico – documento, es obtenida del conteo de las matrices:

$$\phi' = \frac{C^{WK}_{i,k} + \beta_{i,k}}{\sum_{v=1}^{W}\left(C^{WK}_{v,k} + \beta_{v,k}\right)}, \theta' = \frac{C^{DK}_{d,k} + \alpha_{d,k}}{\sum_{j=1}^{K}\left(C^{DK}_{d,j} + \alpha_{d,j}\right)}. \quad (16)$$

3.5 Conclusiones sobre los modelos

La representación de documentos es un paso muy importante en tareas de procesamiento del lenguaje natural. En la segmentación de textos se han utilizado diferentes modelos para dar solución a esta tarea [12, 16]. El modelo espacio vectorial es uno de los más utilizados, principalmente por su eficiencia para determinar similitud entre unidades textuales. Otra ventaja de este modelo es su gran flexibilidad, que consiste en la posibilidad de utilizar distintos esquemas de pesos y distintas funciones de similitud. Hay que destacar que las comparaciones entre unidades textuales se basan, en su esencia, en una comparación léxica entre estas. Este modelo no realiza una comparación semántica, por lo que unidades textuales que traten el mismo tópico con temimos diferentes tendrán un bajo valor de similitud. No obstante la eficacia que se ha obtenido utilizando este modelo ha sido aceptable para muchas aplicaciones.

El modelo LSA surge con el objetivo de superar las dificultades semánticas, generadas por la sinonimia y la polisemia, para ello utiliza una técnica del algebra lineal conocida como SVD. Otra de las bondades de este modelo es la reducción de dimensiones de los vectores. Uno de los problemas que presenta este modelo es la definición de un mecanismo para seleccionar la cantidad de dimensiones, en casos donde los valores propios son muy pequeños. La cantidad de dimensiones debe ser lo suficientemente grande como para reflejar la estructura real de los datos - el contenido conceptual de los documentos - y que, al mismo tiempo, sea relativamente pequeño, para obtener los efectos deseados de la eliminación de ruido. Por otra parte, este modelo no permite que existan más tópicos que documentos, y en una colección de documentos pueden existir más tópicos que documentos, ya que, un documento puede tener más de un tópico latente. Este problema es suavizado cuando se consideran los párrafos para construir la matriz inicial, en lugar de los documentos, pero de igual forma, un párrafo puede pertenecer a más de un tópico, ya que en los documentos pueden existir diferentes mesclas o relaciones entre tópicos.

Por su parte, PLSA al igual que LSA también trata de reflejar el contenido semántico de los documentos. A diferencia del LSA, que utiliza el SVD, este modelo está basado en un modelo generativo conocido como “Modelo de Aspecto”, el cual también hace reducción de dimensiones, ya que el número de variables latentes (tópicos) es menor que el número de términos. Uno de los problemas que presenta este modelo en su aplicación es que, el coste computacional aumenta con el número de documentos que se utilizan en el entrenamiento de dicho modelo, ya que, este estima las distribuciones de los tópicos independientemente para cada documento. Este problema es resuelto en otro de los modelos probabilísticos existentes: LDA. Para resolver este problema de sobreajuste LDA trata la distribución de tópico como una variable aleatoria *K-dimensional* Dirichlet. Así, los parámetros en el modelo no aumentan con el tamaño del corpus de entrenamiento.

Por otra parte, tanto LSA como PLSA, están basados en el supuesto de “saco de palabras”; es decir, que el orden de las palabras es irrelevante. En teoría de probabilidad, este supuesto es equivalente a la propiedad de intercambiabilidad, sin embargo esta propiedad solo es explotada en estos modelos a nivel de las palabras. Es por ello que LDA es un modelo más completo, ya que, partiendo de este supuesto de “saco de palabras” toma en consideración la propiedad de intercambiabilidad tanto para las palabras como para los documentos. Una limitación de LDA es que falla para modelar directamente la relación entre la aparición de temas. En muchos de los textos del corpus, es natural esperar que las ocurrencias de los tópicos latentes subyacentes estén altamente correlacionadas. En el corpus ciencia, por ejemplo, un artículo sobre la genética puede ser probable que sea también acerca de la salud y la enfermedad, pero poco probable que sea también de astronomía de rayos X. En LDA, esta limitación de modelado se deriva de los supuestos de independencia implícita en la distribución de Dirichlet de las proporciones de tópicos. Específicamente, bajo una Dirichlet, los componentes del vector de proporciones son casi independientes, lo que conduce a la suposición de que la fuerte presencia de un tópico no se correlaciona con la presencia de otro. Se dice “casi independiente”, porque los componentes muestran una ligera correlación negativa debido a la restricción de que tienen que sumar uno.

En sentido general, los modelos antes analizados de una forma u otra parten del supuesto de que los términos entre sí son independientes, es decir, no consideran las posibles relaciones entre las palabras. Es bien conocido que esta suposición no es realista, porque las palabras representan conceptos u objetos que pueden ser similares, o que pueden estar relacionados por muchos tipos de asociaciones semánticas. A pesar de esta limitación los modelos LSA, PLSA y LDA reflejan en mayor medida el contenido semántico de los documentos, con un coste computacional superior.

4 Identificación de los límites de tópicos

Después de representadas las unidades textuales se utiliza una medida de similitud para medir la cercanía entre dos vectores, como por ejemplo la medida del coseno. Una vez obtenidos los valores de similitud entre unidades textuales se pasa a identificar los límites de tópicos. En sentido general, los métodos “golosos” en la segmentación utilizan un mecanismo de *sliding-windows*. En la literatura se han utilizado diferentes tipos de ventanas; por ejemplo, en el método propuesto por Hearst (TextTiling) se utilizan dos ventanas (formadas por un conjunto de oraciones) para asignar una puntuación léxica entre ambas [17]. Uno de los problemas del mecanismo utilizado por Hearst, es que, cuando la puntuación léxica decrece notablemente en una zona del texto, se reconoce como un valle y es muy probable que se asigne un límite de segmento, cuando realmente no es así, y lo que está sucediendo es que existe un párrafo corto u otro tipo de interrupción; como, por ejemplo, citas textuales o párrafos que ejemplifiquen una determinada situación que interrumpe una cadena de párrafos cohesionados.

En el trabajo propuesto por Thorsten Brants y autores, al igual que TextTiling, utilizan dos ventanas, formadas por un conjunto de oraciones, para comparar la distancia entre dos bloques de textos. Pero a diferencia de TextTiling, que identifican los límites de segmentos identificando donde ocurre una baja puntuación léxica, la cual es precedida y sucedida por un mayor valor de similitud, estos autores consideran además la diferencia que existe entre el valor de similitud de la posición actual y el valor de

similitud que lo antecede y precede respectivamente, con el objetivo de minimizar el efecto de los párrafos cortos.

A diferencia de estos trabajos, en el 2004 Hernández y Medina proponen un método llamado TextLec, donde utilizan una sola ventana inferior con el objetivo de disminuir la posibilidad de interrumpir la continuidad de un tópico. Esta ventana es utilizada para determinar la cohesión léxica del párrafo en cuestión con los párrafos de su ventana. Similar a lo propuesto en TextLec, en el método ClustSeg los autores utilizan una sola ventana, pero esta ventana no es utilizada para calcular la cohesión léxica entre párrafos, si no que se utiliza para decidir si dos párrafos adyacentes en un grupo pertenecen al mismo segmento.

En sentido general uno de los problemas que presenta la utilización de un mecanismo de *sliding-windows* es la definición del tamaño que debe tener la o las ventanas utilizadas, el cual varía en cada documento entre otras cosas, por estilo de escritura del autor, etc. Una de las debilidades de estos métodos es que son dependientes de este parámetro, el cual es difícil de conocer.

En muchas ocasiones la transición entre tópicos es muy suave, por lo que es muy difícil detectar los límites, mucho más si se van asignando a medida que se recorre el documento. Por tales motivos, varios autores utilizan algoritmos de optimización, con el objetivo de encontrar la solución óptima global, sacrificando la eficiencia del método. El trabajo propuesto por Heinonen combina la técnica de *sliding-windows* con un algoritmo de programación dinámica. En este trabajo el autor utiliza solo una ventana para recorrer el documento y determina para cada párrafo su párrafo más similar dentro de ella, con vista a disminuir el efecto que tienen sobre la segmentación algunos párrafos cortos, tal como sucede en TextTiling. Un inconveniente que presenta este método es que cada párrafo es asociado con el valor de similitud más alto en su ventana, pero no es capaz de identificar si este valor pertenece a un párrafo que se encuentra por encima o por debajo de él, lo que puede provocar que un párrafo sea insertado en un segmento de forma incorrecta. Además, este método tiene otra dificultad, requiere de la especificación de la longitud aproximada de los sub-tópicos, la cual realmente es un valor impredecible y que no suele ser el mismo para todos los sub-tópicos de un texto.

Otro de los métodos que utiliza programación dinámica es el propuesto por Mimi Lu y autores, donde combinan la técnica de *sliding-windows* con un algoritmo de programación dinámica para encontrar la segmentación de menor costo. El costo de una posible segmentación es definida como la suma de todos los costos de los segmentos que forman dicha segmentación. A diferencia de Heinonen, Mimi Lu y autores utilizan dos ventanas para comparar bloques adyacentes. Una de las deficiencias de este método es que, con el objetivo de hacer comparable el costo los segmentos cortos y largos, introducen un paso de normalización, el cual necesita de un parámetro de penalización, que tiene que ser definido y es ajustado empíricamente.

Una de las debilidades que presentan los métodos de segmentación, tanto los “golosos” como los de “optimización” es que son dependientes de un umbral. Existen diferentes tipos de umbrales, tales como: Umbral de cohesión, Umbral de número de segmentos y Umbral de tamaño de tópico. La definición de un umbral es realmente difícil; ya sea para decidir si dos unidades textuales están cohesionadas, para decidir la cantidad de segmentos que tiene un documento o para definir el tamaño de un tópico. Para decidir si dos unidades están relacionadas se han hecho algunas propuestas en la literatura; por ejemplo, en el método TextLec los autores calculan el umbral automáticamente, el cual es la media de los valores de similitud determinados en cada ventana. Similar a esto en ClustSeg los autores utilizan la misma ventana definida por el usuario para el cálculo del umbral, pero en este trabajo los autores consideran además la mediana de los mayores valores de similitud y la mediana de los menores valores de similitud. En sentido general para la identificación de un umbral “optimo” se han hecho muy pocos trabajos y estos estudios se basan en experimentaciones en diferentes corpus.

5 Taxonomía de los métodos de segmentación lineal de texto por tópicos

En esta sección se muestra una taxonomía de los métodos de segmentación lineal por tópico analizados en este trabajo. Además, se describen en más detalle cada uno de estos métodos.

Tabla 1. Taxonomía de los métodos de segmentación de texto por tópico según representación (VSM, LSA, PLSA y LDA) y estrategia de procesamiento golosa (G) o de optimización (O).

Segmentación lineal por tópico											
VSM			**LSA**			**PLSA**			**LDA**		
Métodos	**G**	**O**	**Métodos**	**G**	**O**	**Métodos**	**G**	**O**	**Métodos**	**G**	**O**
TextTiling	x	-	C99	x	-	Brants-2002	x	-	Misra-2009	-	x
Heinonen	-	x	-	-	-	Mimi Lu-2011	-	x	-	-	-
TextLec	x	-	-	-	-	-	-	-	-	-	-
TSF	x	-	-	-	-	-	-	-	-	-	-
ClustSeg	x	-	-	-	-	-	-	-	-	-	-
NClustSeg	x	-	-	-	-	-	-	-	-	-	-

5.1 TextTiling

TextTiling es un algoritmo de segmentación lineal multi-párrafos propuesto por Hearst en 1994, el cual constituye uno de los estudios más completos sobre el tema [17]. Hearst utiliza la cohesión léxica como mecanismo para determinar los límites entre los tópicos. Partiendo del supuesto que si un grupo de términos léxicos o vocabulario se usa durante el curso de la discusión de un sub-tópico y este sub-tópico cambia, entonces una porción significativa del vocabulario cambia también. Este método cuenta con tres etapas: primeramente el pre-procesamiento, una segunda etapa donde dos bloques adyacentes son comparados utilizando la medida del coseno y por último pasa a determinar los límites de tópicos.

En el pre-procesamiento las *stop-words* o palabras vacías son eliminadas, se extraen las raíces léxicas de los términos y por último los términos resultantes se dividen en secuencias o pseudo-oraciones de un tamaño predefinido.

Posteriormente, utilizando la estrategia de *sliding-windows* recorre el texto y para cada posición construye dos bloques formados por un número predefinido de pseudo-oraciones que anteceden y preceden la posición actual. Luego, se representa cada bloque mediante el modelo de espacio vectorial y la similitud entre ellos se calcula usando la medida del coseno, asignando una puntuación de similitud entre estos bloques. Es decir, la puntuación de una posición *i* depende de cuan similares sean las pseudo-oraciones que forman el bloque que precede y sucede la posición *i*.

Para determinar los límites de tópicos Hearst asigna una puntuación de profundidad a cada espacio entre oraciones donde baja la puntuación léxica (valle), utilizando un umbral, el cual calcula promediando las puntuaciones léxicas. Donde una baja puntuación léxica es precedida y sucedida por una alta puntuación léxica se toma como indicador de cambio de vocabulario y por tanto coincide con un cambio de tópico.

5.2 Segmentación lineal de Heinonen

Heinonen, en 1998 propone un método de segmentación lineal multi-párrafos [9]. Este método utiliza un tratamiento de *sliding windows* para recorrer todo el documento, determinando para cada párrafo su párrafo más similar dentro de la ventana. Para determinar los límites de tópicos utiliza un algoritmo de programación dinámica. Este divide el problema de segmentar el texto completo en el sub-problema de segmentar sólo la porción de texto que se forma desde cada párrafo hasta el primer párrafo.

Este método se inicia con una etapa de pre-procesamiento. Luego, con cada párrafo del documento se construye un vector de cohesión $(Cohe_1 \dots Cohe_n)$, donde cada párrafo es relacionado con su valor más alto que se obtuvo dentro de su ventana, la cual está formada por varios párrafos a su alrededor, párrafos por encima y párrafos por debajo.

Para determinar los límites de tópicos Heinonen utiliza un método programación dinámica. El cual considera todas las segmentaciones posibles y determina la de mínimo costo.

5.3 C99

En el 2001, Choi propuso un nuevo método de segmentación lineal por tópico, en el cual combina el modelo LSA con un algoritmo de clustering para identificar los límites de tópico [5]. Este algoritmo está basado en una propuesta del mismo autor, hecha en el año 2000, con la diferencia de que el significado de cada oración es calculado utilizando el espacio reducido generado por el SVD. Otra de las diferencias con su anterior trabajo es que se necesita de una etapa de entrenamiento.

En el entrenamiento del modelo los documentos son pre-procesados. En esta etapa el autor elimina los signos de puntuación y las *stop-words*. Además, divide el documento en oraciones, considerando estas como unidades textuales mínimas. Luego construye una matriz *A*, donde cada elemento es la frecuencia del término *i* en la oración *j*. Estos valores son ajustados según la función *tf-idf* para obtener la matriz *B*.

Una vez obtenida la matriz *B* se le aplica la descomposición en valores singulares $B = U\Sigma V^T$. Donde las columnas de *U* son los vectores propios de BB^T, que no es más que la matriz de semejanza de las palabras. Las primeras *k* columnas de *U* luego se utilizan en el proceso de segmentación.

Para identificar los límites de tópico se construye primeramente una matriz de similitud entre las oraciones de los documentos a segmentar. La similitud entre dos oraciones *i* y *j* se calcula de la siguiente manera:

$$cos(\lambda_i, \lambda_j) = \frac{\sum_k \lambda_{ik} \times \lambda_{jk}}{\sqrt{\sum_k \lambda_{ik}^2 \times \sum_k \lambda_{jk}^2}}, \tag{17}$$

donde λ_i es el significado de la oración *i*, y se determina de la siguiente manera:

$$\lambda_i = \sum_j f_{ij} \times \lambda_k(j)\,. \tag{18}$$

La expresión f_{ij} es la frecuencia del término j en la oración i y $\lambda_k(j)$ es el vector del término j en el espacio reducido, es decir la i-ésima fila de U.

Luego se construye una segunda matriz, llamada matriz de rango. Esta matriz se obtiene sustituyendo cada valor de la matriz de similitud por su rango en la región local, utilizando una máscara de rango de 11x11. El autor llama rango al número de vecinos que tienen un menor valor de similitud en la región local.

Por último se pasa a determinar los límites de tópicos aplicando un proceso de clustering divisivo. Choi define que un segmento de texto puede estar formado por dos oraciones *i*, *j*; las cuales están representadas como un punto a lo largo de la diagonal de la matriz de rango. Al comenzar este proceso, todo el documento se considera como un segmento de texto coherente, posteriormente este es dividido recursivamente en dos segmentos y el punto que maximice a *D* (incidencia interior) es considerado como un límite potencial. Este proceso para cuando el número de segmentos es obtenido o si el usuario pasa este parámetro. Más formalmente, *D* es definida como sigue:

$$D = \frac{\sum_{k=1}^{m} S_k}{\sum_{k=1}^{m} a_k}, \tag{19}$$

donde s_k es la suma de los valores de rango en el segmento k y a_k es el área interior del segmento k.

5.4 TextLec

TextLec, propuesto en el 2007 por Hernández y Medina es otra propuesta de segmentación lineal por tópico para textos científico-técnicos u otros con características similares [7]. Parten del supuesto de que los cambios de vocabularios coinciden con los límites de tópicos. Este método reconoce a los párrafos como las unidades textuales, asumiendo que todas las oraciones que pertenecen a un mismo párrafo tratan el mismo tópico. La representación de los párrafos se basa en el modelo de espacio vectorial y la similitud entre los párrafos es calculada utilizando la medida del coseno, teniendo en cuenta la cantidad de términos que estos tienen en común. TextLec cuenta con tres etapas: primeramente el pre-procesamiento, una segunda etapa donde se hace un control de los párrafos cohesionados más lejos y por último se identifican los límites de los segmentos.

En la etapa de pre-procesamiento eliminan las *stop-words* y lematizan utilizando el TreeTager; un sistema para marcar con etiquetas y extraer el lema de las palabras en un texto.

Posteriormente se define una ventana inferior confeccionada sólo por párrafos por debajo del párrafo que se evalúa. El tamaño de la ventana es entrado por el usuario y este define el tamaño mínimo de un segmento para que sea válido. Para determinar el párrafo que representa la cohesión dentro de la ventana calculan la similitud utilizando la medida del coseno y toman el párrafo cohesionado más lejano. La determinación de si dos párrafos están cohesionados la hacen utilizando un umbral, el cual es la media de los valores de similitud determinados en cada ventana.

Una vez determinado el párrafo cohesionado más lejano forman los segmentos de tópicos, el cual consiste en incluir los párrafos que se encuentran entre el inicio del segmento y el párrafo cohesionado más lejano.

Como TextLec toma como tamaño mínimo para que un segmento sea válido el tamaño de la ventana entrado por el usuario, puede que se obtengan segmentos que no sean válidos (espurios). Por tal motivo hacen una segunda etapa de la segmentación, donde determinan donde incluir estos segmentos espurios, incluyéndolos en su segmento adyacente más similar. Para decidir a qué segmento adyacente es más similar, toman el valor máximo de similitud de algún párrafo del segmento espurio con algún párrafo del segmento adyacente correspondiente.

5.5 TSF

TSF es un algoritmo de segmentación por tópico lineal propuesto por Roman Kern y Michael Granitzer en el 2009 [5]. Parten del supuesto de que la cohesión léxica es un indicador de cambio de tópico en un documento y utilizan una estrategia de *sliding-windows*. Para calcular la similitud entre dos frases *(i, j),* las cuales son representadas por sus vectores de términos ponderados, utilizan la similitud del coseno. Primeramente, los documentos pasan por una etapa de pre-procesamiento, donde se eliminan las *stop-word*. Los autores para identificar las *stop-words* en lugar de utilizar una lista fija de estas, las identifican por la distribución en el documento de los términos; es decir las palabras que tienen una distribución uniforme en el documento se consideran *stop-words* y son eliminadas.

Después de depurar las oraciones de *stop-words* el resto de los términos son ponderados, donde el peso para un término *i* es una combinación de su frecuencia dentro de una oración *j* y el número de documentos de un corpus.

Al igual que TextTiling, construyen para cada posición *i* dos bloques, formados por un conjunto de oraciones que preceden y suceden a la posición actual (B_i^{pre} y B_i^{post}). El tamaño del bloque es uno de

los parámetros introducidos por el usuario, el cual refleja el tamaño mínimo para que un segmento sea detectado como válido.

Para cada uno de los dos bloques calculan los pares de oraciones similares. Luego calculan la media aritmética de los valores de similitud para determinar la similitud interna; este valor es interpretado como el nivel de similitud de las oraciones que rodean la posición actual. Donde μ denota el promedio de pares similares entre los dos Bloques;

$$sim_i^{inner} = \frac{\mu\left(B_i^{pre}, B_i^{pre}\right) + \mu(B_i^{post}, B_i^{post})}{2}. \quad (20)$$

Luego, calculan la semejanza de las oraciones de un bloque con la del otro bloque; esta similitud se promedia para dar la similitud exterior, el cual es un indicador de cuán similar es un bloque respecto al otro.

$$sim_i^{outer} = \mu\left(B_i^{pre}, B_i^{post}\right). \quad (21)$$

Una vez obtenidos estos dos valores de similitud, se determina la disimilaridad entre los dos bloques alrededor de la posición actual.

$$dissimilarity_i = \frac{sim_i^{inner} - sim_i^{outer}}{sim_i^{inner}}. \quad (22)$$

Este valor es positivo si la similitud media entre las frases de ambos bloques es inferior a la media de las similitudes dentro de los bloques, lo cual significa que la posición actual es un verdadero límite de tópico. Si el máximo exterior se alcanza (1) es que la similitud exterior es cero y significa que los bloques no tienen términos en común. Si un valor de disimilitud supera el valor umbral (segundo parámetro entrado por el usuario), la posición correspondiente se marca como límite candidato, y este candidato es seleccionado como límite si no hay mayor valor de disimilitud de las siguientes posiciones de oraciones.

5.6 ClustSeg

ClustSeg, propuesto por Abella y Medina en el 2010 es otra propuesta de segmentación lineal por tópico, que tiene como objetivo identificar límites de tópicos en documentos de texto, utilizando la repetición de términos como mecanismo de cohesión léxica [1]. Este método reconoce a los párrafos como las unidades textuales, asumiendo que el conjunto de todas las oraciones o términos de un mismo párrafo definen el o los tópicos que trata este. La representación de los párrafos se basa en el Modelo de Espacio Vectorial y la cohesión entre estos es calculada utilizando la medida del coseno. La identificación de los límites de tópicos es definida sobre los resultados de la aplicación de un algoritmo de clustering utilizando una estrategia de *sliding windows.*

Por otra parte, ClustSeg se basa en que los cambios de vocabulario en un documento coinciden con los cambios de sub-tópicos, por lo que los párrafos que presentan una cohesión léxica significativa entre sí, en cuanto a los términos léxico que usan, tratan sobre los mismos tópicos. Pero, a diferencia de lo planteado en TextLec y en TextTiling, no importa la distancia que exista entre estos, ya que en un documento el autor puede referirse a un tópico anterior, es decir que un párrafo puede estar relacionado con más de un tópico. Basado en estas hipótesis los autores utilizan un algoritmo de agrupamiento con solapamiento para identificar los límites de tópicos.

Los textos antes de ser segmentados pasan por una etapa de pre-procesamiento. En esta etapa los signos de puntuación, números y las *stop-words* son eliminadas. Las *stop-words* se eliminan utilizando

una lista de estas, compuesta por preposiciones, conjunciones, artículos, etc. Además, se lematiza utilizando el TreeTager.

Después de esta etapa ClustSeg cuenta con tres etapas más. Una primera etapa donde se busca la cohesión entre los tópicos utilizando un algoritmo de clustering con solapamiento. El algoritmo utilizado por los autores es ICSD [19]. Este algoritmo recibe cada párrafo representado como un vector y un umbral (pasado por el usuario) y devuelve un conjunto de grupos, donde cada grupo representa un conjunto de párrafos cohesionados, pertenecientes (probablemente) a un tópico independiente.

Posteriormente en una segunda etapa se pasa a la identificación de los tópicos válidos en cada grupo utilizando una ventana para decidir si dos párrafos adyacentes en un grupo pertenecen al mismo segmento. Cada segmento es obtenido uniendo párrafos adyacentes de acuerdo con la ventana predefinida por el usuario, que además define el tamaño mínimo para que un segmento sea seleccionado como válido.

Como resultado de la etapa anterior se obtiene un conjunto de límites de segmentos de tópicos. Como estos segmentos fueron obtenidos de diferentes grupos, y son solapados, estos segmentos podrían también estar solapados. Por tal motivo en una última etapa ClustSeg concatena todos los segmentos que tienen al menos un párrafo en común.

5.7 NClustSeg

Propuesto por Abella y Medina en el 2012, es una extensión del método propuesto por los propios autores en el 2010 (ClustSeg). Este método principalmente mejora 3 deficiencias del anterior trabajo. Primero, proponen una estrategia para el cálculo automático del umbral, el cual es utilizado por un algoritmo de agrupamiento para formar los conjuntos de párrafos que están en un mismo grupo. Para el cálculo del umbral, utilizan la ventana definida por el usuario y calculan la similitud entre cada par de párrafos, así como el promedio de similitud, el mayor valor de similitud y el menor valor de similitud. Luego se forman 3 set de semejanzas, el primero formado por los promedios de cada ventana, el segundo formado por los mayores valores de similitud y el tercero formado por los menores valores de similitud. Una vez obtenido estos set, se determinan las medianas de cada uno de los set y estos tres valores son promediados, tomando a este como el umbral óptimo.

La segunda mejora que proponen los autores es una estrategia para determinar automáticamente el tamaño mínimo de segmento válido en cada grupo de párrafos cohesionados, obtenidos del algoritmo de agrupamiento. Este tamaño se calcula promediando la diferencia entre cada párrafo (consecutivos) del grupo. Si la diferencia entre un par de párrafos consecutivos es mayor que la ventana definida por el usuario este no es considerado en el promedio y en su lugar se considera el tamaño de la ventana. Además, los autores también calculan un tamaño mínimo para todo el documento, el cual se determina promediando todos los tamaños mínimos de cada grupo, donde al menos existe un segmento válido. Este tamaño mínimo para todo el documento es utilizado para determinar dónde ubicar los segmentos espurios. Los autores definen como segmentos espurios a los párrafos que no se encuentran en ningún segmento, luego de concatenados los segmentos obtenidos de los grupos. Estos pueden ser un segmento válido, en caso que su tamaño sea mayor o igual que el tamaño mínimo para todo el documento. De no ser así estos párrafos son ubicados en el segmento superior o inferior más similar.

5.8 Segmentación lineal de Brants y autores

Thorsten Brants y autores proponen en el 2002 un método de segmentación lineal que se enfoca en identificar sub-tópicos en documentos [4]. En este trabajo los autores combinan el modelo PLSA con la estrategia de comparar la distancia entre dos bloques de textos y seleccionan los límites de tópicos basados en los valores de similitud entre los pares de bloques adyacentes, similar a lo propuesto por Hearst [16].

Durante el pre-procesamiento los autores eliminan los términos con una baja frecuencia, para ello utilizan un umbral. Además, identifican los límites de las oraciones y dividen el texto en bloques, donde cada bloque *b* está formado por un conjunto de oraciones. Los bloques se utilizan en la etapa de entrenamiento para estimar los parámetros del modelo PLSA: $P(w|z)$ y $P(z|b)$, utilizando el algoritmo EM, donde el número de clases latentes *Z* tiene que ser definido (según las experimentaciones hechas por los autores plantean que debe ser dos veces el número de tópicos asignados por jueces humanos). Los valores estimados de $P(w|z)$ luego son utilizados para recalcular los valores de $P(z|b')$ para el documento a segmentar, de igual manera que en la etapa de entrenamiento. Una vez que son obtenidos estos dos valores se utilizan para estimar la distribución de las palabras para cada bloque *b'* del documento a segmentar de la siguiente manera:

$$P(w|b') = \sum_{z} P(w|z)P(z|b') \,. \tag{23}$$

La distribución de las palabras en bloques adyacentes es comparada utilizando una métrica de similitud. Los autores evaluaron la eficacia del método utilizando 5 medidas diferentes, entre las cuales se encontraba la del coseno; sin embargo, el método obtuvo mejores resultados utilizando la métrica *Clarity*.

Una vez obtenidos los valores de similitud entre los bloques adyacentes del documento se identifican los *Dips*. Los autores definen los *Dips* como los mínimos locales en la similitud de los bloques adyacentes; es decir, si una baja puntuación léxica es precedida y sucedida por una alta puntuación léxica (pico) esta es considerada como un *Dips*. Finalmente, para identificar los límites de tópicos calculan a cada *Dips* su tamaño relativo, si este es mayor que un umbral se considera que en dicha posición hay un cambio de tópico, de lo contrario es descartado. El tamaño relativo de un *Dips* se determina mediante la siguiente expresión:

$$d_{rel}(b_l, b_r) = \frac{sim_{lmax}(b_l, b_r) + sim_{rmax}(b_l, b_r)}{2sim_{min}(b_l, b_r)} - 1 \,, \tag{24}$$

donde $sim_{lmax}(b_l, b_r)$ es la diferencia entre el valor de similitud de la izquierda y el valor de similitud del *Dips*, $sim_{rmax}(b_l, b_r)$ es la diferencia entre el valor de similitud de la derecha y el valor de similitud del *Dips* y $sim_{min}(b_l, b_r)$ es el valor de similitud del *Dips*.

5.9 Segmentación lineal de Mimi Lu y autores

Mimi Lu y autores proponen en el 2011 un método de segmentación lineal, el cual combina el modelo PLSA con un algoritmo de programación dinámica para identificar los cambios de historias en transcripciones de noticias [15]. Por otra parte, utilizan la métrica *Cross Entropy* para medir la cohesión léxica entre bloques adyacentes.

Durante el pre-procesamiento los documentos sufren una serie de transformaciones. En la etapa de tokenización, se eliminan las *stop-words* y se lematiza. En un segundo paso se dividen los documentos en bloques. El proceso de formación de bloques se trata de forma diferente para la colección de entrenamiento y para los documentos a segmentar. En la colección de entrenamiento cada bloque *b* representa una historia real (tópico). Mientras que en los documentos a segmentar no es así, ya que no se cuenta con ninguna etiqueta entre historias, ni se tiene información de los límites de oraciones, es por ello que dividen los textos en bloques de tamaño fijo.

Los bloques se utilizan en la etapa de entrenamiento para estimar los parámetros del modelo PLSA: $P(w|z)$ y $P(z|b)$, utilizando el algoritmo EM, donde el número de clases latentes *Z* tiene que ser definido. Los valores estimados de $P(w|z)$ luego son utilizados para recalcular los valores de $P(z|b')$ para el documento a segmentar, de igual manera que en la etapa de entrenamiento. Estos dos valores

son utilizados para estimar la distribución de las palabras para cada bloque *b'*, del documento a segmentar de la siguiente manera:

$$P(w|b') = \sum_{z} P(w|z)P(z|b') \,. \tag{25}$$

La distribución de las palabras en bloques adyacentes es utilizada para medir la diferencia entre dos bloques b_i y b_j, utilizando la medida de divergencia *Cross Entropy*:

$$CrossEnt(i,j) = -\sum_{w} P(w|b_i) logP\big(w\big|b_j\big) \,. \tag{26}$$

Finalmente, la medida de disimilitud es normalizada como:

$$Dissim(i,j) = \frac{CrossEnt(i,j) - CrossEnt(i,i)}{CrossEnt(i,j)}. \tag{27}$$

Una vez obtenidos los valores de disimilitud entre los bloques adyacentes utilizan un algoritmo de programación dinámica para obtener la solución óptima global que mejor capture los cambios de tópicos. Sea $S = \{s_1,\ s_2, \dots, s_k\}$ una hipótesis de segmentación del documento $D = \{b_1,\ b_2, \dots, b_n\}$, dividido en *K* historias. El costo de agrupar un número de bloques en un segmento s_k es representado por las disimilitudes totales entre los bloques en s_k:

$$cost_{u \leftrightarrow v} = cost\ (s_k) = \frac{\sum_{i=u}^{v-1} \sum_{j=i+1}^{v} Dissim(i,j)}{N\big(len(s_k)\big)}, \tag{28}$$

donde *u* y *v* son el primer y último bloque de s_k, y $N\big(len(s_k)\big)$ es un factor de normalización donde $len(s_k)$ es el número de bloques en el segmento s_k. Generalmente la suma de la divergencia toma un valor grande cuando existen más bloques en el segmento. Por lo que, con el objetivo de hacer comparable el costo los segmentos cortos y largos, introducen un paso de Normalización.

$$N(l) = \ l^{\alpha}, \alpha > 1 \,. \tag{29}$$

Donde l es el tamaño del segmento y α funciona como parámetro de represión, el cual es ajustado empíricamente.

El costo total de una posible segmentación *S* es definida como la suma de todos los costos de los *K* segmentos:

$$C(S) = \sum_{k=1}^{K} cost(s_k) \,. \tag{30}$$

Finalmente, es seleccionada la segmentación de menor costo:

$$S = arg_s\, min\, C(s) \,. \tag{31}$$

5.10 Segmentación lineal de Hemant Misra y autores

Hemant Misra y autores proponen en el 2009 un método de segmentación lineal que se enfoca en identificar sub-tópicos en documentos [13]. En este trabajo los autores combinan el modelo LDA con un algoritmo de programación dinámica para identificar los cambios de tópicos en documentos. Parten del supuesto de que, el uso de un modelo de tópico permite una mejor detección de los límites de segmentos, ya que, el cambio de segmento debe estar asociado con un cambio de tópico significativo en la distribución de tópicos.

Durante el entrenamiento se estiman dos sets de parámetros de una colección de documentos: la distribución de tópicos en cada documento d $(\theta_{dt}, t = 1 \dots T, d = 1 \dots D)$ y la distribución de las palabras en cada tópico $(\phi_{tv}, t = 1 \dots T, v = 1 \dots V)$, utilizando el muestreo de Gibbs (Ver ecuación 13). Luego, estos parámetros son utilizados durante el proceso de segmentación.

Para identificar los límites de segmentos, los autores parten del supuesto de que, si un segmento es coherente (la distribución de tópicos para un segmento tiene solo unos pocos tópicos activos), el *log-likelihood* para ese segmento es mayor en comparación con el *log-likelihood* de un segmento que no es coherente. Este valor de *log-likelihood* es utilizado en un algoritmo de programación dinámica para identificar los límites de segmentos.

A continuación se describen los pasos del algoritmo de segmentación (el parámetro ϕ del modelo LDA (distribución de las palabras en cada tópico) se obtuvo durante el entrenamiento de datos):

1. Para cada posible segmento, S_i:
   a. Se calcula Θ realizando 20 iteraciones de:

$$\theta_{S_i t} \leftarrow \frac{1}{n_i} \sum_{v=1}^{V} \frac{C_{w_i v} \theta_{S_i t} \phi_{tv}}{\sum_{t'}^{T} \theta_{S_i t'} \phi_{t'v}}, \tag{32}$$

   donde n_i es la cantidad de términos del segmento.
   b. Se calcula su *log-likelihood* utilizando;

$$P(W_i|\Theta, \Phi) = \prod_{v=1}^{V} \left[\sum_{t=1}^{T} (\theta_{S_i t} \phi_{tv}) \right]^{C_{w_i v}}. \tag{33}$$

   c. La probabilidad de un segmento es considerada como su puntuación;

$$log(W_i|S_i) = logP(W_i|\Theta, \Phi). \tag{34}$$

2. Se sustituyen las puntuaciones de los segmentos en:

$$P(S|d) \propto \left[\prod_{i=1}^{m} \prod_{j=1}^{n_i} P\left(w_i^j | S_i\right) \right] P(S), \tag{35}$$

y se utiliza DP para encontrar la segmentación que maximice la puntuación. El factor de penalización utilizado por los autores fue:

$$logP(S) = -p.m.log(l_d), \tag{36}$$

donde el valor de p se tomó como 3, seleccionado empíricamente.

6 Experimentaciones

Detectar los límites de tópicos en un documento tiene dos problemas fundamentales. Primero, seleccionar el texto de referencia, ya que identificar los límites reales es muy difícil, incluso, para las personas, por la naturaleza subjetiva de los límites de tópicos.

Para dar solución a esta dificultad generalmente lo que se hace es concatenar varios documentos en un fichero y se evalúa cuán bien el método de segmentación distingue un documento de otro. Mientras que otros autores comparan sus resultados contra el resultado de una segmentación manual basada en el juicio de diferentes personas [3, 5, 20].

El segundo problema está dado por la selección adecuada de una medida para evaluar la eficacia de los métodos, ya que depende de la aplicación que tendrá el método.

En la actualidad existen diferentes medidas para evaluar los resultados de la segmentación por tópicos, las cuales serán explicadas a continuación.

6.1 Medidas de evaluación

Existen diferentes medidas para evaluar los resultados experimentales como son: *Precision*, *Recall*, *F-measure*, P_k y *WindowDiff*. A continuación se expondrá en qué consiste cada una de las medidas antes mencionadas.

6.1.1 Precision, Recall y F-measure

Precision y *Recall* son dos medidas muy utilizadas en las experimentaciones con sistemas de recuperación de información, la cuales son utilizadas en la tarea de segmentación de textos para medir la eficacia de esta tarea.

La utilización de estas debe ser en sistemas donde la exactitud de la segmentación sea de vital importancia. Otra dificultad presente en estas medidas es que hay una compensación inherente entre ellas; es decir, cuando una mejora en ocasiones la otra declina. *Precision* puede ser definida como el por ciento o índice que representan los límites de segmento correctamente detectados por el algoritmo del total de límites detectados por el algoritmo. Por otra parte *Recall* es el por ciento o índice que representan los límites de segmento correctamente detectados por el algoritmo del total de límites reales en la segmentación de referencia.

F-measure es otra medida utilizada en la recuperación de información que también se utiliza en la segmentación de textos. La utilización de *F-measure* elimina una de las dificultades presente en *Precision* y *Recall*, pero mantiene la dificultad que penaliza muy fuerte al algoritmo cuando los resultados no son exactos a la segmentación de referencia. *F-measure* es definida como:

$$F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \quad (37)$$

6.1.2 P_k

P_k es una medida propuesta en 1997 y 1999 por Beeferman, Berger y Lafferty, la cual es definida como la probabilidad de que dos oraciones tomadas aleatoriamente del texto sean correctamente clasificadas, como pertenecientes o no al mismo segmento. Sean $sref$ y $salg$, la segmentación de referencia y la segmentación del algoritmo respectivamente, P_k es definida en la ecuación 38 como:

$$P_k(sref, salg) = \sum_{1<i<j<n} D(i,j)\left(\delta_{sref}(i,j) \oplus \delta_{salg}(i,j)\right), \quad (38)$$

donde *n* es el número total de unidades textuales en el documento según el interés de la segmentación. i y j son dos unidades textuales separadas a una distancia k. δ_{sref} es una función que toma el valor 1 ó 0, toma el valor 1 si las unidades textuales i y j pertenecen al mismo segmento en la segmentación de referencia, y toma el valor de 0 en caso contrario. Mientras que δ_{salg} es una función indicador que toma el valor de 1 si las unidades textuales i y j pertenecen al mismo segmento en la segmentación obtenida por el algoritmo, y toma el valor de 0 en caso contrario. El operador $\oplus$ es la función XNOR. La función D_k es una distribución de probabilidad de distancia sobre el conjunto posibles distancias entre las unidades textuales seleccionadas aleatoriamente. Según lo planteado por los autores el valor más adecuado de k se corresponde con la mitad del tamaño promedio de los segmentos en la segmentación de referencia.

Esta medida a diferencia de *Precision* y *Recall* toma en cuenta la proximidad entre los límites de la segmentación de referencia y la segmentación obtenida. A pesar de esto esta medida presenta algunas deficiencias. Una de estas es que penaliza más fuerte el algoritmo cuando ignora un límite de segmento que cuando lo pone incorrectamente.

Otro problema importante con esta medida es que permite que algunos errores no se penalicen [20]. En particular, no toma en consideración el número de límites de segmentos entre los extremos del área de prueba. Ejemplo, sea r_i el número de límites entre los extremos del área de prueba de acuerdo a la segmentación de referencia, y a_i el número de límites propuestos por el algoritmo de segmentación para el mismo intervalo de texto. Si $r_i = 1$ (la segmentación de referencia indica un solo límite) y $a_i = 2$ (el algoritmo marca dos límites en el intervalo de texto), en este momento el algoritmo introduce erróneamente un falso positivo (limite espurio). Sin embargo, la métrica de evaluación P_k no penaliza esta situación. Lo mismo sucede si $r_i = 2$ y $a_i = 1$, el algoritmo en este caso introduce erróneamente un falso negativo (pierde limite), pero tampoco es penalizado por esta medida. En [20] se describen otros problemas que presenta esta medida.

6.1.3 WindowDiff

En el 2000 Pevzner y Hearst proponen una nueva métrica llamada *WindowDiff*, la cual mejora el proceso de evaluación de P_k [20]. *WindowDiff* usa una *sliding-window* de longitud k para procesar todo el documento y encontrar las discrepancias entre la segmentación de referencia y la que se obtiene como resultado del algoritmo de segmentación. Al igual que lo propuesto por Beeferman, Berger y Lafferty, los autores mantienen a k igual a la mitad del promedio del tamaño que tienen los segmentos en la segmentación de referencia.

Para cada posición dentro de la ventana se determina para ambas segmentaciones (la de referencia y la obtenida) el número de límites dentro de la ventana, y si el número de límites no es el mismo se penaliza el algoritmo. Posteriormente, se suman todas las penalizaciones que se encontraron en el texto completo y se normaliza este valor de forma tal que la métrica tome un valor entre 0 y 1. *WindowDiff* toma el valor de 0 si el algoritmo asigna todos los límites correctamente y toma el valor de 1 si difiere con la segmentación de referencia en todas las posiciones de la ventana, por lo que mientras menor sea el valor de *WindowDiff* mayor será el desempeño del algoritmo. Sean $sref$ y $salg$, la segmentación de referencia y la segmentación del algoritmo respectivamente, *WindowDiff* es definida en la ecuación 39 como:

$$WindowDiff(sref, salg) = \frac{1}{N-k}\sum_{i-1}^{N-k}(|b(sref_i, sref_{i+k}) - b(salg_i, salg_{i+k})| > 0)\,, \tag{39}$$

donde $b(i,j)$ representa el número de límites entre la posición i y j en el texto y N representa el número total de unidades textuales en el documento.

Esta medida además de que tiene en cuenta la distancia entre los límites de la segmentación, también considera el número de segmentos en la ventana, penalizando al algoritmo.

6.2 Corpus y análisis experimentales

En las experimentaciones se consideraron todos los métodos para evaluar sus eficacias. En el análisis se tomó el método NClustSeg, desarrollado por los autores, como referencia para su evaluación. Para ello, las experimentaciones se dividieron en dos grupos: sobre textos largos y textos cortos. Ya que el desempeño de los métodos es diferente considerando tales características.

6.2.1 En textos largos

A continuación se muestran algunas de las experimentaciones realizadas por los autores de NClustSeg en corpus largos. Estas experimentaciones fueron publicadas en el RECPAT 2012.

Tabla 2. Corpus utilizados en las experimentaciones.

Nombre	***Descripción***	***Número de Párrafos***
Texto 1	El mismo corpus utilizado por los autores de ClustSeg.	305
Texto 2	Uniendo dos diferentes tópicos de artículos. El "Background" de [21] y "An Introduction to Latent Semantic Analysis" de [16].	22
Texto 3	Este corpus esta formado por el trabajo publicado en [2], eliminando el resumen, conclusiones y referencias.	31
Texto 4	El fichero del corpus TDT2 (19980104_1337_1411_NYT_NYT.sgm).	317
Texto 5	El fichero del corpus (19980601_1201_1217_APW_ENG.sgm).	58
Texto 6	Concatenación de diferentes noticias (AF941122_0126.noti, AF940804_0203.noti, AF940622_0001.noti, AF941121_0103.noti, AF941225_0037.noti, AF941018_0167.noti, AF940524_0311.noti, AF940729_0100.noti, AF941228_0261.noti)	109
Texto 7	Concatenación de diferentes noticias (AF940830_0110.noti, AF941230_0016.noti, AF940911_0024.noti, AF940622_0060.noti)	52

En las siguientes tablas se muestran que la eficacia de NClustSeg es mejor que la del resto de los algoritmos con los que fue comparado. Mostrándose que las mejoras hechas a este método mejoran la eficacia de ClustSeg y la del resto de los algoritmos.

Tabla 3. Valores de WindowDiff en el Texto 1.

Algoritmos	***ClustSeg***	***NClustSeg***	***TextLec***	***TextTiling***	***Heinone's***	***C99***
WindowDiff	*0.11*	*0.07*	*0.21*	*0.33*	*0.26*	*0.21*

Los resultados obtenido por ClustSeg que se muestran en la tabla anterior fueron obtenidos con una ventana igual a 13 y un umbral de 0.35. Sin embargo, el resultado obtenido por NClustSeg fue con valores de ventana desde 8 hasta 14, además sin necesidad de definir un umbral, ya que este es calculado automáticamente. En la tabla 4 se pueden ver los resultados obtenidos por ClustSeg y NClustSeg en el Texto 1 con diferentes valores de ventana.

Tabla 4. Valores de WindowDiff en el Texto 1 con diferentes tamaños de ventana.

Ventana	***Algoritmo***	
	ClustSeg	***NClustSeg***
15	*0.23*	*0.17*
14	*0.11*	*0.07*
13	*0.11*	*0.07*
12	*0.12*	*0.07*
11	*0.13*	*0.07*
10	*0.13*	*0.07*
9	*0.14*	*0.07*
8	*0.14*	*0.07*
7	*0.19*	*0.10*

Los resultados que se ofrecen en la Tabla 5 se realizaron con el objetivo de ver el desempeño de NClustSeg en textos más cortos. Donde se muestra que la eficacia de NClustSeg es superior a la del resto de los métodos.

Tabla 5. Valores de WindowDiff en el Texto 2 y Texto 3.

Corpus	***Algorithm***				
	ClustSeg	***NClustSeg***	***C99***	***TextLec***	***TSF***
Texto 2	*0*	*0*	*0.31*	*0*	*0*
Texto 3	*0.21*	*0.18*	*0.28*	*0.32*	*0.28*

En la tabla anterior se ve que varios métodos identifican correctamente todos los tópicos en el Texto 2. Tanto ClustSeg como TextLec y NClustSeg utilizan un mecanismo de *sliding-windows* para identificar los límites de tópicos y estos identifican correctamente los límites de tópicos. Sin embargo NClustSeg es el único método que identifica correctamente todos los tópicos con más de un valor de ventana. En la tabla 6 se pueden ver dichos resultados.

Tabla 6. Valores de WindowDiff en el Texto 2 variando el tamaño de la ventana.

Window	**Algorithm**		
	ClustSeg	***NClustSeg***	***TextLec***
8	*0.16*	*0*	*0.16*
7	*0.16*	*0*	*0.16*
6	*0.16*	*0*	*0.16*
5	*0.16*	*0*	*0.16*
4	*0*	*0*	*0*
3	*0*	*0*	*0*

Por último se ofrece una tabla con la eficacia de los métodos en los textos formados de los corpus TDT2 y AFP. Donde se muestra que la eficacia de NClustSeg es superior a la obtenida por el resto de los métodos.

Tabla 7. Valores de WindowDiff en Texto 4, 5, 6, 7.

Corpus	***Algoritmos***			
	ClustSeg	***NClustSeg***	***TextLec***	***C99***
Texto 4	*0.17*	*0.16*	*0.27*	*0.16*
Texto 5	*0.22*	*0.07*	*0.22*	*0.30*
Texto 6	*0.18*	*0.15*	*0.25*	*0.49*
Texto 7	*0.19*	*0.04*	*0.11*	*0.32*

6.2.2 En textos cortos

Como se mencionó anteriormente, los modelos LSA, PLSA, LDA necesitan de una fase de entrenamiento, donde se tiene que contar con muchos documentos, los cuales tienen que estar relacionados con los textos que se utilizarán en la segmentación. Uno de los corpus más utilizados en la literatura es el propuesto por Choi [10]. Este corpus está compuesto por noticias, donde cada documento es la concatenación de 10 de segmentos de tópicos y cada segmento está compuesto por *n* oraciones seleccionadas aleatoriamente del corpus Brown.

Para evaluar la eficacia de NClustSeg con los modelos antes descritos se utilizó el corpus de Choi, el cual está dividido en 4 conjuntos de 50 textos cada uno. Para el entrenamiento se utilizaron 3 de estos conjuntos, seleccionados aleatoriamente y se testeo con el restante. A continuación se muestra como se utilizaron estos modelos con NClustSeg:

- NClustSeg + LSA (Se evaluó utilizando la estrategia propuesta por Choi en [10]).
- NClustSeg + PLSA (Se evaluó utilizando la estrategia propuesta por Thorsten Brants y autores en [18]).
- NClustSeg + LDA (Se evaluó utilizando la estrategia propuesta por Hemant Misra y autores en [13]).

Las experimentaciones demostraron que NClustSeg no es eficaz en documentos muy cortos y con muy baja cohesión, considerando directamente los lemas o raíces de los términos. Haciendo un análisis del porqué, se observa como posible causa la aplicación del algoritmo de agrupamiento seleccionado considerando una representación VSM. Para trabajos futuros se recomienda experimentar con métodos de agrupamiento de textos cortos y la consideración de otros modelos de representación orientados a estos tipos de textos. A continuación se muestra una tabla con los resultados de algunos métodos publicados por los autores en dicho corpus [13]. La eficacia de los métodos se midió utilizando la métrica P_k, donde a medida que el valor sea menor, mejor es el desempeño del método.

Tabla 8. Valores de P_k en el repositorio propuesto por Choi.

Algoritmo	P_k ***en %***			
	3-5	***6-8***	***9-11***	***3-11***
C99 (2000)	*12*	*9*	*9*	*12*
C99 (2001)	*10*	*7*	*5*	*9*
LDA [13]	*2.2*	*2.3*	*4.1*	*2.3*

Según lo reportado por Hemant Misra y autores en [13], su método, el cual utiliza el modelo LDA obtiene mejores resultados, no obstante, se precisan otras experimentaciones para tener criterios más sólidos.

7 Conclusiones

La segmentación de texto es una tarea importante la cual puede mejorar los resultados de diferentes tareas de procesamiento de textos como son: la recuperación de información, la generación automática de resúmenes, entre otras.

En la actualidad, aunque existen varios métodos de segmentación la eficacia de estos no es del todo aceptable, ya que usualmente generan segmentos incompletos o espurios. Además, pocos consideran la mezcla de tópicos en los segmentos que descubren, o las diferentes relaciones que estos presentan. Por tales motivos, la investigación del tema sigue siendo de interés con el objetivo de encontrar nuevas soluciones, las cuales mejoren los resultados obtenidos hasta el momento y que no tengan un alto coste computacional.

Uno de los principales pasos en la segmentación de texto es la selección de un modelo de representación, el cual debe considerar tanto la información sintáctica como semántica de los documentos. En la actualidad aunque existen algunos modelos que tratan en alguna medida ambos aspectos, todavía no son capaces de representar las diferentes relaciones semánticas que existen entre los términos. Además, estos modelos tienen un alto coste computacional, que es aún mayor cuando es combinado con algoritmos de optimización.

En el presente trabajo se analizaron diferentes modelos que se han utilizado en la segmentación de texto, así como los principales métodos que utilizan dichos modelos. Para trabajos futuros se recomienda analizar otros modelos. Además las experimentaciones mostraron que el método NClustSeg obtiene mejor eficacia que el resto en textos largos; sin embargo su eficacia es mala en textos cortos, aspecto que debe evaluarse para lograr mejores desempeños.

8 Referencias bibliográficas

1. Abella-Pérez, R. and J.E. Medina-Pagola, *Text Segmentation by Clustering Cohesion.* CIARP 2010, LNCS 6419, Springer-Verlag Berlin Heidelberg, 2010: p. 261-268.
2. Abella-Pérez, R. and J.E. Medina-Pagola, *An Incremental Text Segmentation by Clustering Cohesion.* The International Workshop on Handling Concept Drift in Adaptive Information Systems: Importance, Challenges and Solutions (HaCDAIS 2010). Workshop del ECML-PKDD 2010. , 2010: p. 65-72.
3. Hernández, L. and J.E.M. Pagola, *TextLec: A Novel Method of Segmentation by Topic Using Lower Windows and Lexical Cohesion.* CIARP 2007, 2007: p. 724–733.
4. Hernández, L. and J.E.M. Pagola, *TextLec: Método para la Segmentación por Tópicos en Textos Científicos-Técnicos.* Reporte Técnico Minería de Datos, Serie Gris, CENATAV, 2009.
5. Ken, R. and M. Granitzer, *Efficient Linear Text Segmentation Based on Information Retrieval Techniques.* MEDES 2009 Lyon, France, 2009.
6. Shi, Q., et al., *Semi-Markov Models for Sequence Segmentation.* Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2007: p. 640–648.
7. Stokes, N., J. Carthy, and A. Smeaton, *SeLeCT: A Lexical Cohesion Based News Story Segmentation System.* AI Communications 17(1), 2004: p. 3–12.
8. Bolshakov, I. and G. A, *Text segmentation into paragraphs based on local text cohesion.* 2001.
9. Heinonen, O., *Optimal Multi-Paragraph Text Segmentation by Dynamic Programming.* Proceedings of OLING-ACL 1998, Montreal, Canada, 1998: p. 1484–1486.
10. Choi, F.Y.Y., *Latent semantic analysis for text segmentation.* Proceedings of EMNLP, 2001: p. 109–117.
11. Hofmann, T., *Probabilistic Latent Semantic Indexing.* SIGIR '99 8/99 Berkley, CA USA, 1999. **ACM 1-58113-096-1/99/0007**.
12. Lu, M., et al., *Probabilistic Latent Semantic Analysis for Broadcast News Story Segmentation.* 28-31 August 2011, Florence, Italy, 2011: p. 1109-1112.
13. Misra, H., et al., *Text Segmentation via Topic Modeling: An Analytical Study.* Hong Kong, China, 2009.
14. K-Landauer, T., P. W.-Foltz, and D. Laham, *An Introduction to Latent Semantic Analysis.* Discourse Processes, 1998: p. 259-284.
15. Blei, D.M., A.Y. Ng, and M.I. Jordan, *Latent Dirichlet Allocation.* Journal of Machine Learning Research 2002. **3 (2003) 993-1022**.
16. Landauer, T.K., P.W. Foltz, and D. Laham, *An Introduction to Latent Semantic Analysis.* 1998: p. 259-284.
17. M, H., *TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages.* Computational Linguistics, vol, 1997. **23**.
18. Brants, T., F. Chen, and I. Tsochantaridis, *TopicBased Document Segmentation with Probabilistic Latent Semantic Analysis.* CIKM'02, November 4–9, 2002, McLean, Virginia, USA., 2002. **ACM 1581134924/02/0011**.
19. Pérez-Suáerez, A., J.F. Martinez-Trinidad, and J.A. Carrasco-Ochoa, *A New Incremental Algorithm for Overlapped Clustering*. Bayro-Corrochano, E., Eklundh, J.-O. (eds.) CIARP 2009. LNCS. Springer, Heidelberg 2009. **5856**.

20. Pevzner, L. and M. Hearst, *A Critique and Improvement of an Evaluation Metric for Text Segmentation.* Computational Linguistics, vol, 2000.
21. Jia, Y., et al., *Towards comprehensive structural motif mining for better fold annotation in the "twilight zone" of sequence dissimilarity.* The Seventh Asia Pacific Bioinformatics Conference (APBC 2009) Beijing, China, 2009.

RT_025, mayo 2014
Aprobado por el Consejo Científico CENATAV

Editor: Lic. Lucía González Bayona
Diseño de Portada: Di. Alejandro Pérez Abraham
RNPS No. 2143
ISSN 2072-6260
Indicaciones para los Autores:
Seguir la plantilla que aparece en www.cenatav.co.cu
C E N A T A V
7ma. A No. 21406 e/214 y 216, Rpto. Siboney, Playa;
La Habana. Cuba. C.P. 12200
Impreso en Cuba