Universidad Nacional Autónoma de México
Posgrado en Ciencia e Ingeniería de la Computación

Muestreo para modelos generativos de tópicos basado en Min-Hashing

Tesis

que para optar por el grado de
Maestro en Ciencia e Ingeniería de la Computación

presenta:
José Morales Zorrilla

Director de tesis:
Dr. Gibrán Fuentes Pineda
IIMAS

Ciudad Universitaria, Ciudad de México. Mayo 2021

Índice general

Notación

a	Un escalar
$\mathbf{a}$	Un vector
a_i	El i-ésimo elemento del vector $\mathbf{a}$
a_{-i}	Todos los elementos del vector $\mathbf{a}$ excepto el i-ésimo
$\mathbf{A}$	Una matriz
$A_{i,j}$	Elemento i, j de la matriz $\mathbf{A}$
$c_{z,d,w}$	Número de veces que la palabra w ha sido asignada al tópico z en el documento d
$c_{z,*,*}$	Número de palabras asignadas al tópico z
$c_{z,d,*}$	Número de palabras asignadas al tópico z en el documento d
$c_{z,*,w}$	Número de veces que la palabra w ha sido asignada al tópico z
$c_{*,d,*}$	Número de palabras en el documento d
$c_{z,d,w}^{-(a,b)}$	Número de veces que la palabra w ha sido asignada al tópico z en el documento d sin tomar en cuenta la posición (a, b)

Glosario

CGS	Collapsed Gibbs Sampling
LDA	Latent Dirichlet Allocation
LSH	Locality-Sensitive Hashing
MCMC	Markov Chain Monte Carlo
MHGS	Min-Hashed Gibbs Sampling
NPMI	Normalized Pointwise Mutual Information
PMI	Pointwise Mutual Information

Capítulo 1

Introducción

1.1. Motivación

A lo largo de las últimas décadas la información generada por la humanidad ha ido incrementando exponencialmente. En septiembre 2020, Wikipedia tenía 1,627,934 artículos en español, 6,162,065 en inglés y 29,054 millones en total, cubriendo 309 diferentes idiomas. En octubre 2017, PubMed (una herramienta de búsqueda de literatura biomédica) contenía información bibliográfica de más de 27 millones de artículos. De manera análoga, el número de artículos publicados en arXiv (un servicio de distribución de artículos científicos de acceso abierto) por mes ha crecido exponencialmente en la última década, como se muestra en la figura 1.1.

De forma paralela al crecimiento en el volumen de datos se han desarrollado técnicas para poder recolectar, procesar, almacenar y analizar dicha información. Un campo que ha cobrado mucha importancia es la minería de textos, la cual busca extraer información útil de grandes colecciones de documentos utilizando generalmente métodos automáticos.

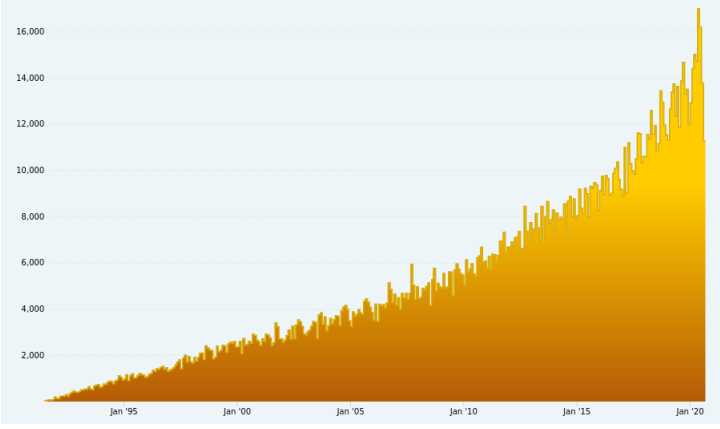

Figura 1.1: Número de artículos públicados en arXiv por mes. Fuente `https://arxiv.org/stats/monthly_submissions`

El modelado de tópicos es una técnica de minería de textos que provee herramientas para ayudarnos a entender la estructura de una colección de archivos electrónicos. Un tópico se puede pensar como un conjunto de palabras relacionadas entre sí y que se pueden encontrar frecuentemente en el mismo documento. El modelado de tópicos busca encontrar estos tópicos en colecciones de documentos, identificar cuáles están presentes en cada uno y con qué proporción. Una vez obtenida esta información se puede utilizar para organizar, resumir e indexar cada documento para posteriores búsquedas.

El modelado de tópicos puede ser utilizada también para estudiar la prevalencia de los tópicos a lo largo del tiempo, así como analizar correlaciones entre tópicos y permite hacer conjeturas sobre fenómenos ocurridos en la historia. D. J. Newman y Block (2006) analizan la discusión política en la Gaceta de Pensilvania durante el periodo previo a la revolución de Estados Unidos (1760s).

Las técnicas que se han desarrollado para el modelado de tópi-

cos pueden ser aplicadas también a campos diferentes a la minería de textos, ya que su estructura define mezclas sobre variables latentes discretas. L. Liu, Tang, Dong, Yao, y Zhou (2016) presentan un análisis de las aplicaciones que han tenido en el campo de la bioinformática, las cuales incluyen clasificación, agrupamiento y generación de características para entrenar otros modelos. En este campo los tópicos se vuelven procesos biológicos, los cuales son mezclas de palabras biológicas.

El modelado de tópicos también ha sido utilizada en el campo de visión computacional. Sivic, Russell, Efros, Zisserman, y Freeman (2005) encuentran objetos en imágenes viendo a una imagen como una mezcla de objetos y crean el equivalente visual a las palabras utilizando el descriptor SIFT cuantizado. Russell, Freeman, Efros, Sivic, y Zisserman (2006) obtienen varias segmentaciones de cada imagen en una colección y utilizan la modelación de tópicos para encontrar agrupaciones de segmentaciones similares. Wang y Grimson (2008) desarrollaron un modelo llamado *Spatial Latent Dirichlet Allocation* que incorpora la estructura espacial inherente en los equivalentes visuales de las palabras, lo cual permite tomar en cuenta no solo la co-ocurrencia de las palabras (visuales) en un mismo documento (imagen) sino también su proximidad espacial al asociarlas a un mismo objeto.

1.2. Planteamiento del problema

Dentro del modelado de tópicos, *Latent Dirichlet Allocation* (Blei, Ng, y Jordan, 2003) es la técnica más popular. *Latent Dirichlet Allocation* (LDA) es un modelo cuyos supuestos son que cada documento es una mezcla de tópicos, es decir, cada documento contiene diferentes proporciones de cada tópico (las cuales suman 1) y cada tópico es una mezcla de palabras. El proceso de inferencia de este modelo consiste en estimar estas proporciones, por lo que

una vez realizada la inferencia se cuenta con una aproximación de la proporción de cada tópico en cada documento y de cada palabra en cada tópico.

La principal dificultad que presenta el modelo LDA es la estimación de sus parámetros, ya que la solución exacta es intratable, por lo que todos los métodos utilizados para estimarlos utilizan aproximaciones y, dependiendo del enfoque, presentan diferentes limitaciones. Las principales limitaciones en las diferentes soluciones para estimar los parámetros de LDA son el tiempo que toma estimar los parámetros del modelo, uso de memoria, escalamiento a datos masivos, distribución de cómputo, entre otros.

Una de las metodologías más populares para realizar la inferencia de los parámetros de LDA es el *Collapsed Gibbs Sampling* (CGS), ya que su implementación es muy fácil y eficiente en memoria. Sin embargo, la convergencia puede ser lenta ya que el muestreo se realiza sobre cada palabra de cada documento y la información que se provee al proceso es solo la tupla (documento, palabra), a partir de la cual se determinan las probabilidades de que pertenezca a cada uno de los tópicos.

La técnica Min-Hashing Broder, Charikar, Frieze, y Mitzenmacher (1998) permite agrupar elementos similares y es comúnmente utilizada para encontrar vecinos cercanos aproximados en espacios con muchas dimensiones. En el caso de minería de textos, ésta genera colecciones de palabras que ocurren frecuentemente juntas y probablemente pertenezcan a un mismo tópico.

1.3. Hipótesis

La hipótesis en que se basa este trabajo es que los conjuntos formados por Min-Hashing pueden ser integrados en el proceso de muestreo de manera que se realice una actualización en bloque,

considerando todas las variables incluidas en cada conjunto. En teoría ambos enfoques convergen al mismo resultado con suficientes iteraciones. Conforme el tamaño de los conjuntos se acerca a uno, la convergencia del método propuesto se aproxima a la del CGS original.

Se presupone que cada conjunto generado por Min-Hashing pertenece a un solo tópico y el problema consiste en encontrar qué tópico es al que pertenece. Esto para evitar la explosión combinatoria que se genera al intentar asignar cada elemento en la muestra a cualquiera de los tópicos como la que se muestra en X. Zhang y Sisson (2016), así como realizar de manera más efectiva la asignación de tópicos

1.4. Objetivos

1.4.1. Generales

Diseñar y desarrollar un algoritmo de CGS para modelos de tópicos que saque provecho de las colecciones creadas por Min-Hashing, de manera que se pueda realizar una actualización en bloque.

1.4.2. Particulares

- Determinar la forma en que se debe aplicar la técnica Min-Hashing de forma exhaustiva a un corpus, de manera que cada elemento sea asignado a una de las colecciones generadas por Min-Hashing.
- Analizar el proceso de muestreo del CGS y definir los cálculos necesarios para poder calcular la probabilidad de que cada

una de las colecciones generadas por Min-Hashing pertenezca a cada uno de los tópicos.

- Desarrollar un algoritmo que reciba un corpus, asigne cada palabra a una colección utilizando Min-Hashing, y realice la inferencia de los parámetros de LDA utilizando el CGS modificado para realizar actualizaciones en bloque.
- Evaluar la efectividad del algoritmo propuesto tomando en cuenta velocidad de convergencia, calidad de los resultados y tiempo de ejecución.

1.5. Contribuciones

Las contribuciones del presente trabajo son las siguientes:

- Un procedimiento para pasar de la representación de matriz documento-término a una de cubetas de palabras a través de Min-Hashing, con la posibilidad de realizarse en paralelo para acelerar la ejecución al tener corpus con muchos documentos.
- Un nuevo algoritmo para realizar la inferencia aproximada de los parámetros del modelo LDA. Este algoritmo podría reemplazar directamente al CGS ya que funciona de manera análoga pero explotando de mejor manera la información al realizar las actualizaciones de Gibbs por bloques en lugar de elementos individuales.
- Evaluación del algoritmo propuesto utilizando los corpus Twenty News Groups y Reuters, contrastándolo contra el CGS original en tiempo de ejecución, coherencia de los tópicos al término de cada algoritmo, así como la log-verosimilitud obtenida por iteración.

1.6. Organización

La estructura del trabajo es la siguiente: En el capítulo 2 se realiza una breve revisión del estado del arte, mencionando los diferentes enfoques que se han utilizado para realizar la inferencia de los parámetros del modelo LDA. Posteriormente, en el capítulo 3 se provee el marco teórico necesario para definir el método propuesto. En el capítulo 4 se describe el procedimiento para obtener los conjuntos y cómo se modifica el proceso de muestreo para asignar un mismo tópico a cada conjunto, realizando así una actualización en bloque. En el capítulo 5 se presentan los resultados del algoritmo propuesto midiendo tiempo de ejecución, log-verosimilitud por iteración y coherencia al final de las iteraciones contrastándolos con el algoritmo base. Finalmente, en el capítulo 6 se incluyen conclusiones de la propuesta, áreas de oportunidad y posible trabajo futuro.

Capítulo 2

Marco teórico

2.1. Representación de documentos

En el procesamiento de textos es común utilizar una representación conocida como *bolsa de palabras*, en donde se presupone que cada documento se puede representar como un conjunto de palabras, en donde no importa el orden en que aparecen las palabras, sino solo su frecuencia. Este supuesto permite representar un documento como un vector en donde cada palabra tiene un índice asignado y cuyo valor en esa entrada corresponde a la frecuencia con la que aparece dicha palabra. Es decir, si a la palabra *hola* se le asigna el índice 20 y esta palabra aparece 5 veces en el documento, entonces $v_{20} = 5$.

Si consideramos un corpus, es decir, un conjunto de documentos, entonces, bajo el supuesto de la bolsa de palabras, éste puede ser representado como una *matriz documento-término*, en donde cada renglón corresponde a un documento y cada columna a un término o palabra. Por ejemplo, si denominamos A a la matriz documento-término, entonces A_{ij} contiene la frecuencia de la palabra asignada

al índice j en el documento i. Esta matriz generalmente es muy dispersa y no suele almacenarse completa sino solo las entradas mayores a cero.

2.2. Modelación de tópicos

Cuando se tiene un archivo de documentos muy grande, usualmente se desea organizar, entender, buscar o resumir los documentos de manera automática. El modelado de tópicos permite etiquetar cada documento con los tópicos que contiene, así como identificar las palabras más relevantes en cada tópico. Posteriormente se pueden utilizar dichas etiquetas para organizar y buscar documentos similares, así como descubrir los patrones contenidos en un documento específico. Los métodos más efectivos para modelar tópicos son aquellos basados en enfoques probabilísticos, como los presentados por (Blei, 2012).

2.2.1. Latent Dirichlet Allocation

La distribución de Dirichlet es una distribución continua multivariada de dimensión k, que está parametrizada por un vector de k números reales positivos $(\theta_1, \theta_2, \ldots, \theta_k) \sim Dir(\alpha_1, \alpha_2, \ldots, \alpha_k)$ donde $k \geq 2$ y $\alpha_i > 0 \ \forall i$. Una propiedad útil de esta distribución es que los valores θ se encuentran en el $(k-1)$ símplex, es decir, $\sum_{i=1}^{k} \theta_i = 1$ y $\theta_i \geq 0 \ \forall i$.

La distribución de Dirichlet es también la distribución a priori conjugada de la distribución multinomial; es decir, si $\Theta \sim Dir(\alpha)$ y $X \sim Multinomial(\Theta, N)$ con $N \in \mathbb{Z}^+$, entonces $\Theta|X$ también se distribuye Dirichlet.

En el dominio de análisis de textos, *Latent Dirichlet Allocation* (LDA), propuesto por Blei y cols. (2003), es un modelo bayesiano

jerárquico en donde cada documento se modela como una mezcla sobre tópicos y cada tópico se modela como una mezcla sobre un vocabulario. La representación en placas del modelo se muestra en la figura 2.1.

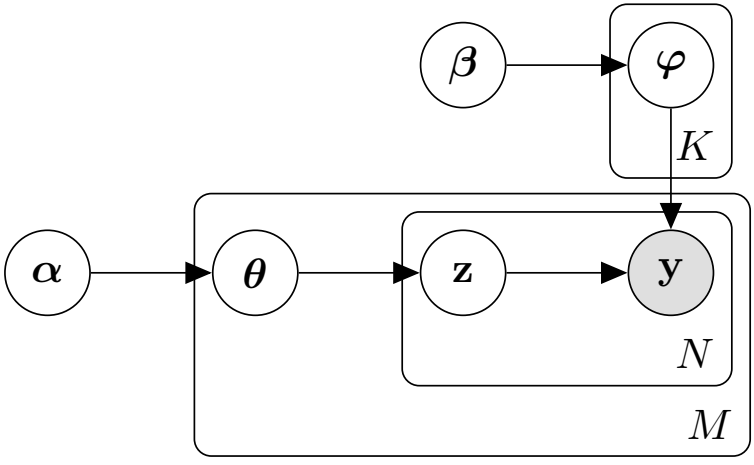

Figura 2.1: Representación en notación de placas de LDA.

Donde:

- K es el número de tópicos.
- M es el número de documentos en el corpus.
- N es la longitud del documento.
- $\boldsymbol{\varphi} \sim Dirichlet(\boldsymbol{\beta})$ modela la distribución de palabras sobre tópicos.
- $\boldsymbol{\theta} \sim Dirichlet(\boldsymbol{\alpha})$ modela la distribución de tópicos sobre documentos.
- $z_{m,n}$ es el tópico de la palabra $y_{m,n}$.

En la representación de la figura 2.1 podemos apreciar que la única variable observable es y. α y β son hiperparámetros del modelo, generalmente definidos como escalares, lo que genera distribuciones simétricas para las variables φ y θ.

Una vez que se tienen las distribuciones φ y θ, el proceso generativo es el siguiente:

- Para cada documento $d = 1, 2, \ldots, M$.
  - Para cada elemento $n = 1, 2, \ldots, N$.
    - Elegir un tópico $z_{d,n} \sim Multinomial(\theta_d, 1)$.
    - Elegir una palabra $y_{d,n} \sim Cat(\varphi_{z_{d,n}})$.

El mayor obstáculo a superar en este modelo es la inferencia de sus parámetros (φ y θ), ya que no es posible calcularlos exactamente, por lo que usualmente se recurre a métodos que permitan aproximarlos. Una alternativa es utilizar Gibbs Sampling (GS). Para este modelo, la distribución objetivo es (2.1).

$$\begin{aligned} P(\mathbf{y}, \mathbf{z}, \theta, \varphi | \alpha, \beta) = &\prod_{k=1}^{K} P(\varphi_k|\beta) \prod_{d=1}^{M} P(\theta_d|\alpha) \\ &\times \prod_{n=1}^{N} P(z_{d,n}|\theta_d) P(y_{d,n}|\varphi_{z_{d,n}}) \end{aligned} \tag{2.1}$$

En la práctica se utiliza una variante llamada *Collapsed Gibbs Sampling*, descrita por Griffiths y Steyvers (2004), en donde las distribuciones de Dirichlet (θ y φ) se expresan como distribuciones marginales y se eliminan, lo que facilita el proceso de muestreo, ya que la distribución objetivo se convierte en $P(\mathbf{y}, \mathbf{z}|\alpha, \beta)$. Luego, el

algoritmo muestrea la asignación de cada tópico condicionando en las demás asignaciones, como se muestra en (2.2)

$$P(z_{d,n} = k|z_{-(d,n)}, \mathbf{y}, \alpha, \beta) \propto (c_{k,d,*}^{-(d,n)} + \alpha)\frac{c_{k,*,y_{d,n}}^{-(d,n)} + \beta}{c_{k,*,*}^{-(d,n)} + V\beta} \quad (2.2)$$

Donde:

- V es el tamaño del vocabulario.
- $z_{d,n} \in \{1, \ldots, K\}$ es el tópico asignado al n-ésimo elemento del documento d.
- $y_{d,n} \in \{1, \ldots, V\}$ es la palabra correspondiente al n-ésimo elemento del documento d.
- $c_{k,d,w}^{-(d,n)}$ es el número de veces en que la palabra w ha sido asignada al tópico k en el documento d sin tomar en cuenta el n-ésimo elemento del documento d. $*$ denota una suma sobre esa posición, por ejemplo $c_{k,d,*}^{-(d,n)}$ representa el número de elementos en el documento d que han sido asignados al tópico k sin tomar en cuenta la posición n.

2.3. Gibbs Sampling

Gibbs Sampling es una técnica *Markov Chain Monte Carlo* (MCMC) cuyo objetivo es generar muestras de una distribución multivariada para la cual es difícil obtener muestras directamente pero cuyas distribuciones condicionales son fáciles de simular.

El procedimiento consiste en:

1. Obtener las distribuciones condicionales de cada variable de manera individual.

2. Inicializar cada variable con valores aleatorios.

3. Generar muestras barriendo sobre cada variable para muestrear de la distribución condicional con las demás variables fijas en sus valores actuales.

Si se repite el procedimiento anterior suficientes veces, la distribución converge a la distribución objetivo.

Generalmente se eliminan las primeras muestras, ya que éstas no provienen de la distribución objetivo y se extraen muestras espaciadas para reducir la correlación que existe entre las muestras consecutivas.

En el capítulo 4 se profundiza más sobre el papel de GS en el proceso de estimación de los parámetros de LDA.

2.4. Min-Hashing

Locality Sensitive Hashing (LSH) es una técnica que permite encontrar vecinos aproximados en espacios de muchas dimensiones, como describen Slaney y Casey (2008b). En particular, esta técnica permite tomar un elemento y encontrar otros elementos similares a él con un costo pequeño en comparación con la alternativa de fuerza bruta y con una cierta probabilidad de obtener falsos positivos o falsos negativos que puede ser disminuida agregando cómputo.

Min-Hashing es un esquema de LSH en el que se definen funciones *hash* de manera que la probabilidad de que a un par de conjuntos S_i, S_j se les asigne el mismo valor bajo la función h sea

igual a su similitud de Jaccard (2.3).

$$P(h(S_i) = h(S_j)) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = J(S_i, S_j) \tag{2.3}$$

Lo anterior significa que la probabilidad de que dos conjuntos reciban el mismo valor bajo la función h es el cociente entre el tamaño de su intersección y el tamaño de su unión. Por lo tanto dos conjuntos idénticos tienen probabilidad 1 de recibir el mismo valor bajo h y dos conjuntos ajenos tienen probabilidad cero de recibir el mismo valor.

En análisis de textos, a partir de una matriz documento-término se puede calcular el valor Min-Hash para un término (una columna de la matriz) permutando los renglones y encontrando el primer renglón que no es cero. En la práctica este enfoque es muy costoso, por lo que existen diferentes técnicas para no permutar la matriz directamente, por ejemplo las descritas por Rajaraman y Ullman (2011).

Usualmente se calculan $M = r \cdot l$ valores Min-Hash para cada conjunto y se construyen l tablas en donde cada cubeta se define con una tupla de r valores Min-Hash. Cada tabla particiona el espacio de conjuntos de manera que conjuntos similares tienen una alta probabilidad de estar en la misma cubeta en cada una de las tablas. Concretamente, para dos conjuntos S_i, S_j:

1. La probabilidad de que compartan un valor Min-Hash es $J(S_i, S_j)$.

2. La probabilidad de que sean asignados a la misma cubeta en una tabla es $J(S_i, S_j)^r$.

3. La probabilidad de que no sean asignados a la misma cubeta en una tabla es $1 - J(S_i, S_j)^r$.

4. La probabilidad de que no sean asignados a la misma cubeta en ninguna de las tablas es $(1 - J(S_i, S_j)^r)^l$.

5. La probabilidad de que sean asignados a la misma cubeta en al menos una de las tablas es $1 - (1 - J(S_i, S_j)^r)^l$.

Esta técnica se puede utilizar directamente para el modelado de tópicos, como muestran Fuentes-Pineda y Meza-Ruiz (2019).

2.5. Pointwise Mutual Information

Newman, Lau, Grieser, y Baldwin (2010) describen métricas para evaluar la coherencia de los tópicos obtenidos por técnicas automáticas de modelado de tópicos. *Pointwise Mutual Information* (PMI) es una medida de asociación entre parejas de realizaciones de variables aleatorias discretas a través de la razón de la probabilidad conjunta y el producto de las probabilidades condicionales y se define en la ecuación 2.4.

$$PMI(x, y) = \log\left(\frac{P(x, y)}{P(x)P(y)}\right) \tag{2.4}$$

donde $P(x, y)$ es la probabilidad conjunta y $P(x), P(y)$ son las probabilidades marginales.

La PMI se puede normalizar al rango [-1, 1], utilizando la ecuación 2.5 y se conoce como *Normalized Pointwise Mutual Information* (NPMI).

$$NPMI(x, y) = \frac{PMI(x, y)}{-\log(P(x, y))} = \frac{\log(P(x)P(y))}{\log(P(x, y))} - 1 \tag{2.5}$$

Donde:

- $NPMI(x, y) = -1$ significa que x y y jamás ocurren juntas.
- $NPMI(x, y) = 0$ significa que x y y ocurren juntas por casualidad.
- $NPMI(x, y) = 1$ significa que x y y siempre ocurren juntas.

Esta métrica se utiliza para evaluar la coherencia de los tópicos obtenidos utilizando las n palabras más relevantes de cada tópico. Lau y Baldwin (2016) recomiendan calcular la coherencia utilizando diferentes valores de n para cada tópico y reportar el promedio.

Capítulo 3

Estado del arte

Existen varias técnicas para el modelado de tópicos; sin embargo, la más relevante hasta el momento ha sido *Latent Dirichlet Allocation* (LDA). El mayor obstáculo en este modelo es la inferencia de sus parámetros, ya que la inferencia exacta es intratable y se debe recurrir a aproximaciones.

3.1. Métodos de estimación de parámetros para LDA

Desde su introducción, se han desarrollado muchas alternativas para realizar la inferencia de sus parámetros, las cuales se pueden agrupar en dos principales rubros: basados en optimización y basados en muestreo.

3.1.1. Métodos basados en optimización

Los métodos basados en optimización generalmente emplean inferencia variacional, donde el problema de optimización consiste en encontrar una distribución paramétrica que sea cercana (respecto a la divergencia Kullback-Leibler) a la distribución posterior.

En el trabajo de Blei y cols. (2003), donde se introduce LDA, la inferencia se lleva a cabo utilizando inferencia variacional. Teh, Newman, y Welling (2006) propusieron *Collapsed Variational Inference*, en donde se propone aplicar inferencia variacional a la distribución conjunta con las distribuciones de Dirichlet marginalizadas, logrando una mejor aproximación que el método original. Sin embargo, este enfoque no escala bien con los datos.

Online LDA, propuesto por Hoffman, Blei, y Bach (2010), resuelve el problema de la escalabilidad. Esta técnica se basa en optimización estocástica en línea, la cual permite realizar la inferencia con datos masivos, ya que cada documento individual puede ser visto una vez y luego descartado. Foulds, Boyles, DuBois, Smyth, y Welling (2013) desarrollaron una extensión natural al algoritmo *Online LDA*, en donde se aplica la optimización estocástica en línea a la distribución colapsada.

Mimno, Hoffman, y Blei (2012) combinan *Sparse Gibbs Sampling*, que permite realizar la inferencia sobre un gran número de tópicos (realizando un muestreo más eficiente a través de una modificación al cálculo de la constante de normalización y realizando menos actualizaciones en cada mini-batch), con la inferencia estocástica en línea, la cual permite realizar inferencia sobre corpus con muchos documentos.

Recientemente, las redes neuronales profundas han sido utilizadas en cada vez más campos. En el modelado de tópicos se han planteado diferentes soluciones basadas en aprendizaje profundo para resolver algunos de los problemas observados en las soluciones

descritas anteriormente.

Srivastava y Sutton (2017) utilizan *Autoencoding Variational Bayes* realizando una modificación al algoritmo original de LDA en donde la mezcla de palabras sobre tópicos no está restringida al $(k-1)$ simplex, sino que está representada por un producto ponderado de expertos, logrando una mayor coherencia en los tópicos obtenidos.

Miao, Yu, y Blunsom (2016) introducen un marco de trabajo para utilizar redes neuronales en la inferencia variacional en modelos generativos de texto. La propuesta es utilizar una red neuronal para aproximar las distribuciones sobre las variables latentes.

Miao, Grefenstette, y Blunsom (2017) presentan una alternativa basada en redes neuronales en donde se reemplaza la distribución de Dirichlet para los tópicos por una gaussiana que está representada por una red neuronal condicionada por otra gaussiana.

H. Zhang, Chen, Guo, y Zhou (2020) desarrollaron un método que combina el gradiente estocástico MCMC y un autocodificador bayesiano variacional. Este se compone de una red neuronal generativa compuesta de una jerarquía de distribuciones gamma y una red neuronal para realizar la inferencia, la cual es un autocodificador variacional Weibull.

Rezaee y Ferraro (2020) definen una red neuronal recurrente con una capa LSTM que modela la asignación de los tópicos para cada palabra incorporando variables discretas.

3.1.2. Métodos basados en muestreo

Los métodos basados en muestreo utilizan técnicas de *Markov Chain Monte Carlo* (MCMC). El método más popular ha sido el *Collapsed Gibbs Sampling*, propuesto por Griffiths y Steyvers (2004). Esta técnica es la más fácil de implementar y la que re-

quiere la menor cantidad de memoria, por lo que es ampliamente utilizada. Sin embargo, suele ser difícil diagnosticar su convergencia. Otra preocupación es que por su naturaleza, este algoritmo es secuencial, lo que dificulta distribuir la inferencia.

Con el fin de escalar el proceso de inferencia a datos masivos (tanto en documentos como en tópicos y tamaño de vocabulario) se han trabajado formas de distribuir el proceso de muestreo. D. Newman, Asuncion, Smyth, y Welling (2009) proponen relajar el supuesto de que el algoritmo debe realizar las actualizaciones de las frecuencias observadas en los tópicos de manera secuencial, mostrando que dado que el número de palabras es generalmente mucho mayor al número de procesos que se ejecutan de forma concurrente, no existe una fuerte dependencia entre las frecuencias de un tópico con las de los demás. Esto permite realizar la inferencia de manera distribuida y lograr una convergencia más rápida.

La distribución del proceso de muestreo, además de las dificultades teóricas estadísticas presenta también dificultades computacionales, tales como la comunicación entre nodos y la memoria compartida. Z. Liu, Zhang, Chang, y Sun (2011) proponen un esquema en donde se separan las tareas limitadas por CPU y las limitadas por comunicación. Para mitigar el tiempo de comunicación, la información para una palabra se recibe al mismo tiempo que se ejecuta el muestreo sobre otra, con lo cual se logra enmascarar el tiempo de comunicación.

Yuan y cols. (2015) proponen distribuir el conjunto de entrenamiento sobre un cluster de manera que éstos nunca se muevan y el modelo se particiona y almacena de forma distribuida y se envía a los nodos sobre demanda. También introducen un algoritmo de muestreo Metropolis-Hastings más eficiente, así como una estructura de representación híbrida que se adapta a altas y bajas frecuencias en las palabras, logrando una mayor eficiencia en memoria y alto desempeño.

En general cada enfoque tiene ventajas y desventajas, tales como definición de hiperparámetros, uso de memoria, velocidad de convergencia, entre otros. Para tener un panorama más amplio sobre las diferencias y la calidad de los resultados de cada algoritmo, el trabajo de Asuncion, Welling, Smyth, y Teh (2009) contrasta los diferentes algoritmos y presenta resultados empíricos de su desempeño.

3.2. Técnicas basadas en Locality-Sensitive Hashing

Locality-Sensitive Hashing (LSH) se propuso originalmente para aproximar similitudes en parejas entre elementos en conjuntos de datos masivos (Slaney y Casey, 2008a). LSH ha sido generalizado utilizando funciones de *kernel* con diferentes similitudes para realizar búsqueda de imágenes (Kulis y Grauman, 2009) y también para realizar búsqueda aproximada basada en el máximo producto interno (Shrivastava y Li, 2014). Además de esto, ha resultado ser útil en distintas aplicaciones, tales como estimar la constante de normalización en modelos log-lineales Spring y Shrivastava (2017) y realizar *split-merge MCMC* de manera más eficiente Luo y Shrivastava (2019).

Capítulo 4

Descripción del método

Un tópico está compuesto de palabras que pueden encontrarse frecuentemente en el mismo documento y, por tanto, se pueden entender dentro de un mismo contexto. Las palabras más representativas de cada tópico son las que aparecen con mayor frecuencia en los documentos que contienen a ese tópico.

Un método para estimar las distribuciones de tópicos sobre documentos y de palabras sobre tópicos es el Collapsed Gibbs Sampling. Este método consiste en recorrer cada ocurrencia de cada palabra de cada documento para asignarle un tópico y posteriormente aproximar las distribuciones a partir de frecuencias. La información que utiliza para determinar a qué tópico puede pertenecer cada elemento de un documento es qué tan común es cada tópico en ese documento y qué tan común es la palabra asociada a ese elemento en cada uno de los tópicos; este procedimiento se describe en el Algoritmo 1.

La técnica Min-Hashing se puede aplicar sobre una colección de documentos, de tal forma que a cada palabra se le asigne un valor Min-Hash (que estará asociado a un documento que la contiene).

Algoritmo 1: Collapsed Gibbs Sampling

Para cada token $r_d^{(w)}$ del corpus:

Calcular la probabilidad p_k de que pertenezca al tópico k

$p_{k,d}$ = proporción del tópico k en el documento d.

$p_{w,k}$ = proporción de la palabra w en el tópico k.

$p_k = p_{k,d} \cdot p_{w,k}$

Muestrear un tópico $\hat{k} \sim \boldsymbol{p}$ y asignarlo al elemento actual.

Este procedimiento permite agrupar palabras que ocurren juntas frecuentemente y, probablemente, pertenecen a un mismo tópico.

Para obtener un valor Min-Hash a partir de una matriz documento término, se obtiene una permutación de los documentos y se recorre la matriz por columnas (las cuales representan las palabras del vocabulario) buscando en cada una la primer entrada que no es cero, siguiendo el orden de la permutación calculada.

Por ejemplo, si tuviéramos tres documentos y un vocabulario de cuatro palabras, la matriz documento-término podría verse como la siguiente:

$$\begin{matrix} d_1 \\ d_2 \\ d_3 \end{matrix} \begin{bmatrix} 0 & 3 & 4 & 2 \\ 2 & 1 & 0 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}$$

En este ejemplo, una posible permutación es (3, 1, 2), bajo la cual la matriz se vuelve:

$$\begin{matrix} d_3 \\ d_1 \\ d_2 \end{matrix} \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 3 & 4 & 2 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

En la primera columna, el primer valor no cero se encuentra en el primer renglón, por lo que se elige el documento asociado al primer renglón bajo la permutación, el cual corresponde al documento 3. En la segunda columna, el primer valor no cero se encuentra en el segundo renglón y corresponde al documento 1. Para las siguientes dos palabras los renglones serían 2 y 1, correspondientes a los documentos 1 y 3. Finalmente, los valores Min-Hash serían (3, 1, 1, 3).

Una vez que se ha obtenido este valor, se puede utilizar como una llave y todas las palabras con la misma llave son agrupadas. Las palabras en un mismo grupo tras aplicar este método tienen la propiedad de que aparecen en un mismo documento (que es la llave) y probablemente poseen algunas otras similitudes.

En el ejemplo anterior, los grupos serían:

- 3: [1, 4]
- 1: [2, 3]

Dado que un valor Min-Hash puede no ser suficiente, normalmente se generan tuplas de valores Min-Hash, donde mientras más grande es el tamaño de la tupla existe una menor probabilidad de agrupar palabras que no sean similares y que ocurran en el mismo documento por casualidad. Con un tamaño de tupla suficientemente grande se logra que los grupos generados por la técnica Min-Hashing estén compuestos en su mayoría por un mismo tópico.

Continuando con el ejemplo, se podría generar un valor Min-Hash adicional y crear los grupos usando dos valores como llave. Los valores Min-Hash actuales son: (3, 1, 1, 3). Dado que las combinaciones (documento, palabra) [(3, 1), (1, 2), (1, 3), (3, 4)] ya han sido asignadas, restamos 1 a las entradas de la matriz documento-término. Es decir, la matriz ahora sería:

$$\begin{matrix} d_1 \\ d_2 \\ d_3 \end{matrix} \begin{bmatrix} 0 & 2 & 3 & 2 \\ 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Otra posible permutación es (2, 3, 1), la cual genera la matriz:

$$\begin{matrix} d_2 \\ d_3 \\ d_1 \end{matrix} \begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 2 & 3 & 2 \end{bmatrix}$$

Bajo esta permutación, los valores Min-Hash son: (2, 2, 1, 3). Ahora cada palabra tiene asignada una tupla de valores Min-Hash, es decir, para la primer palabra se obtuvo el valor 3 con la primera permutación y 2 con esta nueva permutación, por lo que ahora la primer palabra tiene asociada la tupla (3, 2). De esta forma, los valores Min-Hash para las palabras son: [(3, 2), (1, 2), (1, 1), (3, 3)] y la matriz documento-término es ahora:

$$\begin{matrix} d_1 \\ d_2 \\ d_3 \end{matrix} \begin{bmatrix} 0 & 2 & 2 & 2 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Si quisiéramos asignar una tupla de cuatro valores Min-Hash a cada palabra, nos restarían dos valores Min-Hash por calcular para cada palabra. Otra posible permutación es (1, 3, 2), la cual genera la matriz:

$$\begin{matrix} d_1 \\ d_3 \\ d_2 \end{matrix} \begin{bmatrix} 0 & 2 & 2 & 2 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Los valores Min-Hash para esta permutación son: (3, 1, 1, 1) y la matriz es ahora:

$$\begin{matrix} d_1 \\ d_2 \\ d_3 \end{matrix} \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Finalmente, dado que ya solo queda un valor no cero en cada columna, los últimos valores Min-Hash son (2, 1, 1, 1).

De esta forma, los valores Min-Hash asociados a cada palabra son [(3, 2, 3, 2), (1, 2, 1, 1), (1, 1, 1, 1), (3, 3, 1, 1)]. Si se utilizan

estos valores como llaves y se agrupan las palabras bajo la misma llave, los grupos serían:

- (3, 2, 3, 2): [1]
- (1, 2, 1, 1): [2]
- (1, 1, 1, 1): [3]
- (3, 3, 1, 1): [4]

En este caso cada palabra queda aislada en un grupo y cada grupo contiene 4 elementos del corpus (uno por cada documento en la tupla). En general, cada grupo tiene (tamaño de tupla) · (palabras en el grupo) elementos.

Utilizando el supuesto de que cada grupo creado está compuesto principalmente de un solo tópico, podemos recorrer los grupos generados por Min-Hashing y utilizar las palabras obtenidas en cada grupo para determinar el tópico al que corresponden.

Con lo anterior en mente, el método propuesto en este trabajo consiste en aplicar Min-Hashing a un corpus hasta que cada elemento del corpus haya sido asignado a un grupo. Una vez hecho esto, se supone que cada grupo corresponde a un mismo tópico y se utiliza la información de cada grupo para determinar el posible tópico al que corresponde.

4.1. Cálculo de probabilidades

El proceso descrito en la sección anterior genera cubetas de la forma $(d^{(1)}, d^{(2)}, ..., d^{(r)})$: $[w^{(1)}, w^{(2)}, ..., w^{(n)}]$, donde r es el tamaño de la tupla. Para realizar la inferencia iteramos sobre cada documento en la llave y todas las palabras en los valores, es decir, las muestras son de la forma $s = \{(d, w^{(1)}), (d, w^{(2)}), ..., (d, w^{(n)})\}$.

Siguiendo los pasos en Carpenter (2010), la probabilidad conjunta del modelo LDA es:

$$p(\mathbf{y}, \mathbf{z}|\alpha, \beta) \propto A \times B \tag{4.1}$$

Donde:

$$A = \prod_{m=1}^{M} \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \frac{\prod_{k=1}^{K} \Gamma(c_{k,m,*} + \alpha_k)}{\Gamma(\sum_{k=1}^{K} c_{k,m,*} + \alpha_k)} \tag{4.2}$$

$$B = \prod_{k=1}^{K} \frac{\prod_{w=1}^{V} \Gamma(c_{k,*,w} + \beta_j)}{\Gamma(\sum_{w=1}^{V} c_{k,*,w} + \beta_j)} \tag{4.3}$$

La ecuación (4.2) representa la probabilidad asociada a los documentos y la ecuación (4.3) representa la probabilidad asociada a las palabras, donde $c_{k,d,w} = \sum_{n=1}^{N} \mathbb{1}(z_{d,n} = k, y_{d,n} = w)$.

4.1.1. Probabilidad asociada al documento

Partiendo de (4.2), si solo el documento d es observable, entonces

$$A \propto \frac{\prod_{k=1}^{K} \Gamma(c_{k,d,*} + \alpha_k)}{\Gamma(\sum_{k=1}^{K} c_{k,d,*} + \alpha_k)}$$

Definimos $c_{k,d,w}^{-s}$ como el número de veces en que la palabra w ha sido asignada al tópico k en el documento d sin tomar en cuenta los elementos en s.

Suponemos que todos los elementos en la muestra provienen del mismo tópico k_s, entonces $c_{k,d,*} = c^{-s}_{k,d,*}\ \forall k \neq k_s$ y $c_{k_s,d,*} = c^{-s}_{k_s,d,*} + |s|$, entonces:

$$A \propto \frac{\prod_{k \neq k_s} \Gamma(c^{-s}_{k,d,*} + \alpha_k) \times \Gamma(c^{-s}_{k_s,d,*} + \alpha_k + |s|)}{\Gamma(|s| + \sum_{k=1}^{K} c^{-s}_{k,d,*} + \alpha_k)}$$

Notemos que

$$\Gamma(c^{-s}_{k_s,d,*} + \alpha_{k_s} + |s|) = \prod_{j=0}^{|s|-1} (c^{-s}_{k_s,d,*} + \alpha_{k_s} + j)\Gamma(c^{-s}_{k_s,d,*} + \alpha_{k_s})$$

Como el denominador es constante se puede eliminar:

$$A \propto \prod_{j=0}^{|s|-1} (c^{-s}_{k_s,d,*} + \alpha_k + j)$$

Si suponemos $\alpha_k = \alpha\ \forall k$,

$$A \propto \prod_{j=0}^{|s|-1} (c^{-s}_{k_s,d,*} + \alpha + j)$$

4.1.2. Probabilidad asociada a las palabras

Partiendo de (4.3) quitamos lo que no está en la muestra:

$$B \propto \prod_{k=1}^{K} \frac{\prod_{w \in s} \Gamma(c_{k,*,w} + \beta_j)}{\Gamma(\sum_{w=1}^{V} c_{k,*,w} + \beta_j)}$$

Sea $s_w = \{(d, w) \in s\}$, entonces $c_{k_s,*,w} = c^{-s}_{k_s,*,w} + |s_w|$.

$$B \propto \prod_{k \neq k_s}^{K} \frac{\prod_{w \in s} \Gamma(c^{-s}_{k,*,w} + \beta_w)}{\Gamma(\sum_{w=1}^{V} c^{-s}_{k,*,w} + \beta_w)} \times \frac{\prod_{w \in s} \Gamma(c^{-s}_{k_s,*,w} + \beta_w + |s_w|)}{\Gamma(|s| + \sum_{w=1}^{V} c^{-s}_{k_s,*,w} + \beta_w)}$$

Notemos que

$$\Gamma(c^{-s}_{k_s,*,w} + \beta_w + |s_w|) = \prod_{i=0}^{|s_w|-1} (c^{-s}_{k_s,*,w} + \beta_w + i)\Gamma(c^{-s}_{k_s,*,w} + \beta_w)$$

y

$$\Gamma\left(\sum_{w=1}^{V}(c^{-s}_{k_s,*,w} + \beta_w) + |s|\right) = \prod_{j=0}^{|s|-1}\left(\sum_{w=1}^{V}(c^{-s}_{k_s,*,w} + \beta_w) + j\right) \times \Gamma\left(\sum_{w=1}^{V}(c^{-s}_{k_s,*,w} + \beta_w)\right)$$

Entonces

$$B \propto \frac{\prod_{w \in s} \prod_{i=0}^{|s_w|-1}(c^{-s}_{k_s,*,w} + \beta_w + i)}{\prod_{j=0}^{|s|-1}\left(\sum_{w=1}^{V}(c^{-s}_{k_s,*,w} + \beta_w) + j\right)}$$

Suponemos $\beta_w = \beta \ \forall w$, por lo que

$$B \propto \frac{\prod_{w \in s} \prod_{i=0}^{|s_w|-1}(c^{-s}_{k_s,*,w} + \beta + i)}{\prod_{j=0}^{|s|-1}\left(c^{-s}_{k_s,*,*} + V\beta + j\right)}$$

4.1.3. Probabilidad de la muestra

Finalmente, a partir de (4.1) la probabilidad de que todos los elementos en la muestra pertenezcan a un tópico k se muestra en la ecuación (4.4)

$$p(\mathbf{y}, \mathbf{z}|\alpha, \beta) \propto \prod_{j=0}^{|s|-1} \frac{c_{k,d,*}^{-s} + \alpha + j}{c_{k,*,*}^{-s} + V\beta + j} \times \prod_{w \in s} \prod_{i=0}^{|s_w|-1} (c_{k,*,w}^{-s} + \beta + i) \quad (4.4)$$

Si solo hay un elemento w en la muestra, es decir, $s = \{(d, w)\}$, entonces (4.4) se puede reescribir como (4.5).

$$(c_{k,d,*}^{-s} + \alpha) \frac{c_{k,*,w}^{-s} + \beta}{c_{k,*,*}^{-s} + V\beta} \quad (4.5)$$

que es igual a la probabilidad que utiliza el CGS (2.2).

4.2. Algoritmo

Algoritmo 2: Min-Hashed Gibbs Sampling

Convertir la matriz documento-término a la representación de cubetas.
Para cada cubeta calculada:
  Para cada documento en la llave:
    si *la cubeta contiene más de una palabra* **entonces**
      Calcular la probabilidad de que pertenezca a cada uno de los tópicos usando la ecuación (4.4)
      Muestrear un tópico $\hat{k} \sim \boldsymbol{p}$ y asignarlo a todos los elementos en la cubeta.
    en otro caso
      Para cada ocurrencia de la palabra en la cubeta:
        Calcular la probabilidad de que pertenezca a cada uno de los tópicos usando la ecuación (4.5).
        Muestrear un tópico $\hat{k} \sim \boldsymbol{p}$ y asignarlo al elemento actual.

Capítulo 5

Evaluación

Para evaluar la efectividad del algoritmo propuesto se hizo una comparación contra el Collapsed Gibbs Sampling estándar en los corpus Twenty Newsgroups y Reuters. Los corpus de Twenty Newsgroups y Reuters contienen 11,314 y 806,791 documentos respectivamente. En cada caso se quitaron *stopwords* y se realizó una lematización. Adicionalmente, se eligieron los 20,000 términos más frecuentes como vocabulario.

Para ambos corpus se fijaron los hiperparámetros de LDA en $\frac{1}{\text{número de tópicos}}$. Para Twenty Newsgroups se buscaron 20 tópicos y para Reuters 50.

En el caso de Twenty Newsgroups se realizaron 100 experimentos ejecutando cada algoritmo por 50 iteraciones y para Reuters se realizaron 5 experimentos de 100 iteraciones. Con el propósito de observar el impacto del tamaño de la tupla en MHGS se experimentó con tamaños de tupla de 3 a 8.

5.1. Metodología

La evaluación consiste en medir la velocidad de convergencia, la calidad de los resultados y la eficiencia de cada algoritmo.

Para evaluar la convergencia, en cada iteración se calculó la log-verosimilitud, la cual mide qué tan plausible es que el modelo con los parámetros actuales haya generado los datos observados.

La calidad de los tópicos se evaluó utilizando su coherencia, la cual mide el grado de similitud semántica entre las palabras clasificadas como importantes dentro de cada tópico. Específicamente, se utilizó la *Normalized Pointwise Mutual Information* (NPMI) promediada para las 5, 10, 15 y 20 palabras principales de cada tópico.

La eficiencia se evaluó utilizando el tiempo para realizar la primera iteración, el tiempo promedio por iteración y el tiempo total de ejecución. Para el corpus de Reuters se observó que el tiempo para realizar el preprocesamiento era considerablemente mayor que el de Twenty Newsgroups, esto debido a que los valores Min-Hash se buscan sobre los documentos (renglones), por lo que al tener más documentos el algoritmo debe realizar más iteraciones para poder determinar el valor Min-Hash de cada palabra. Esto se mitigó particionando la matriz documento-término horizontalmente, de manera que se pudiera recorrer un menor número de renglones en cada partición y el preprocesamiento pudiera llevarse a cabo en paralelo. Se generaron 64 particiones de los 806,791 renglones, de manera que cada partición tuvo aproximadamente 12,600 renglones y se utilizaron 8 hilos para llevar a cabo el preprocesamiento en paralelo.

Los experimentos se realizaron utilizando el lenguaje de programación Julia (Bezanson, Edelman, Karpinski, y Shah (2017)). La implementación del CGS contra la que se comparó fue la de

TextAnalysis.jl[1].

5.2. Resultados

5.2.1. Convergencia

En la figura 5.1 se muestra la velocidad de convergencia de cada algoritmo en Twenty Newsgroups. Se observa que con menores tamaños de tupla el algoritmo propuesto converge más rápidamente que el CGS y para mayores tamaños de tupla la diferencia se reduce. Esto es consistente con el hecho de que si cada cubeta tiene solo un elemento, los algoritmos son iguales.

En la figura 5.2 se muestran los resultados para los cinco experimentos realizados en Reuters. En este corpus la diferencia entre las dos implementaciones no es tan dramática como en Twenty Newsgroups y conforme aumenta el tamaño de tupla se vuelven casi indistinguibles.

[1] `https://github.com/JuliaText/TextAnalysis.jl`

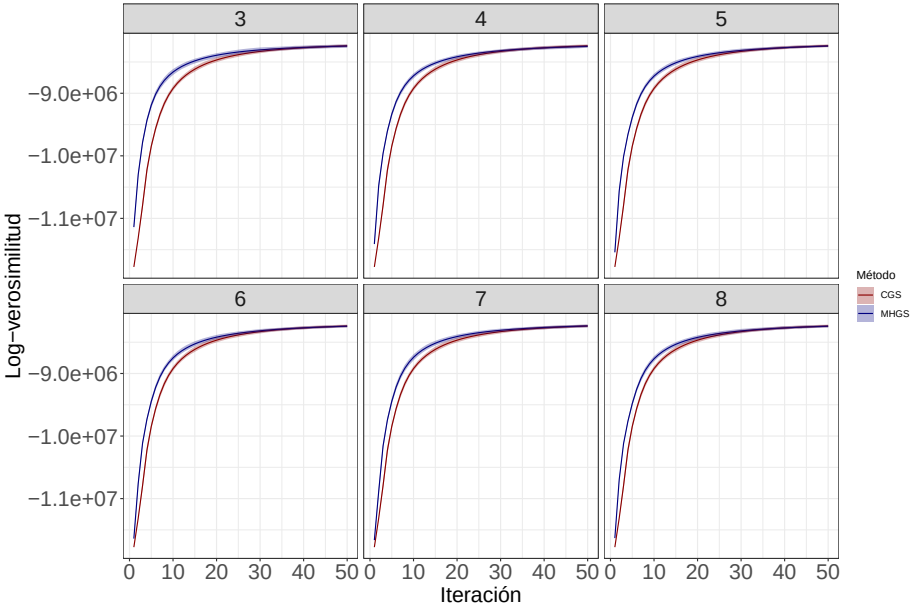

Figura 5.1: Twenty News Groups
Log-verosimilitud contra iteración por tamaño de tupla.
El tamaño de tupla se muestra en la parte superior de cada gráfica.

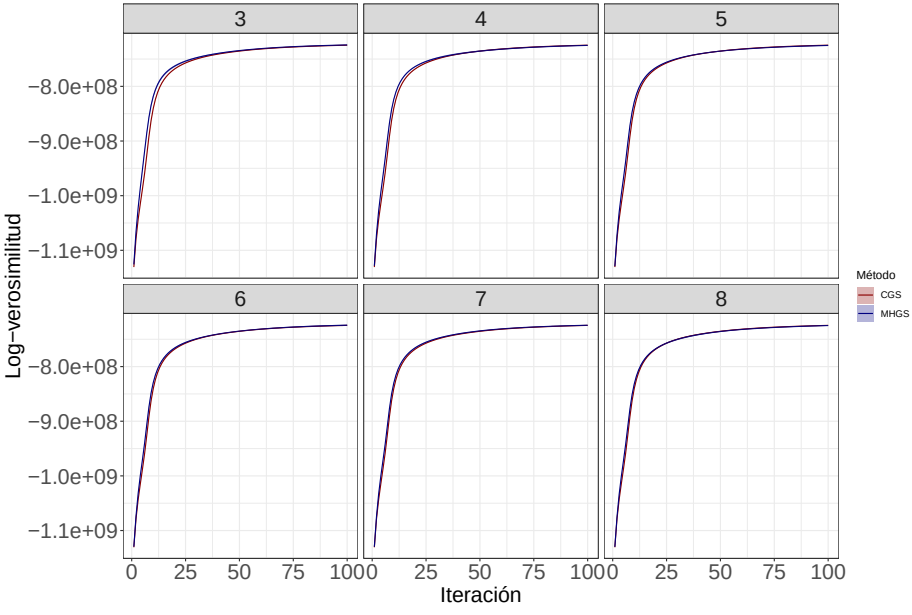

Figura 5.2: Reuters
Log-verosimilitud contra iteración por tamaño de tupla.
El tamaño de tupla se muestra en la parte superior de cada gráfica.

5.2.2. Calidad de tópicos

Dado que la coherencia mide el grado de similitud semántica entre las palabras clasificadas como importantes dentro de cada tópico, buscamos maximizarla, por lo que decimos que un método es mejor que otro si logra una mayor coherencia.

En la figura 5.3 se muestran las distribuciones de la coherencia calculada en Twenty Newsgroups. Los triángulos corresponden al promedio y las líneas centrales a la mediana. Podemos apreciar que para tamaños de tupla pequeños (3, 4) a pesar de que la mediana es similar a la del CGS, tanto la media como el tercer cuartil (el extremo superior de las cajas) son inferiores. Conforme aumenta el tamaño de tupla estos estadísticos se aproximan al CGS.

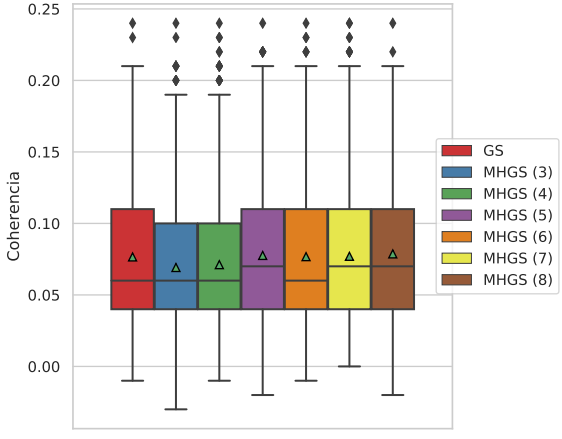

Figura 5.3: Twenty News Groups
Coherencia por método

En la figura 5.4 se muestran las distribuciones de la coherencia calculada en Reuters. En este caso para tamaños de tupla pequeños (3, 4) se logra un tercer cuartil más alto respecto a CGS y conforme aumenta el tamaño de tupla el método se aproxima al CGS. Esto

puede atribuirse al tamaño del corpus en Reuters, ya que las colecciones generadas por Min-Hashing contienen más palabras y las actualizaciones se realizan con un volumen mayor de información.

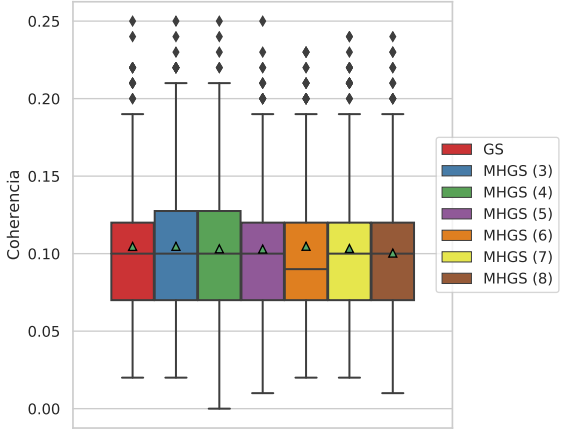

Figura 5.4: Reuters
Coherencia por método

5.2.3. Eficiencia

En la tabla 5.1 se muestran los tiempos de ejecución en el corpus Twenty Newsgroups. El tiempo a la primera iteración incluye el preprocesamiento, por lo que CGS es más rápido siempre (ya que no realiza preprocesamiento). Podemos observar que conforme aumenta el tamaño de tupla el tiempo por iteración disminuye para MHGS, esto se debe a que a mayor tamaño de tupla se generan cubetas más pequeñas y el cálculo de las probabilidades de cada cubeta es más rápido.

En la tabla 5.2 se muestran los tiempos de ejecución en el corpus Reuters. En este caso el preprocesamiento se realizó en parale-

Tabla 5.1: Tiempos en segundos para Twenty Newsgroups. Para MHGS el tamaño de tupla se muestra entre paréntesis.

	Primera iteración	Por iteración	Total
CGS	2.4	1.3	65.8
MHGS (3)	4.8	1.3	68.5
MHGS (4)	4.4	1.2	62.2
MHGS (5)	4.2	1.1	59.2
MHGS (6)	4.1	1.1	56.7
MHGS (7)	4.0	1.0	54.8
MHGS (8)	4.0	1.0	53.4

lo dividiendo la matriz documento-término en 64 subconjuntos de aproximadamente 12,500 documentos y usando 4 hilos.

5.2.4. Ejemplos de tópicos hallados

Tabla 5.2: Tiempos en minutos para Reuters.
Para MHGS el tamaño de tupla se muestra entre paréntesis.

	Primera iteración	Por iteración	Total
CGS	5.5	3.5	182.4
MHGS (3)	21.1	5.2	277.5
MHGS (4)	32.3	4.7	262.7
MHGS (5)	26.2	4.4	245.4
MHGS (6)	22.9	4.3	235.7
MHGS (7)	20.8	4.1	220.4
MHGS (8)	17.4	4.0	212.7

CGS	**MHGS**
space nasa launch system satellite	space launch nasa program satellite
god jesus christian people believe	god christian jesus believe know
game team play season hockey	game team year play good
window file edu image program	window file use program entry
drive card disk scsi mb db	drive card system scsi disk

Tabla 5.3: Ejemplo de tópicos hallados para Twenty Newsgroups

CGS	**MHGS**
israel israeli palestinian peace minister	israel israeli palestinian peace arab
china kong hong chinese taiwan	china hong kong chinese taiwan
game win run play second	win game play match cup
share company million stock percent	company share million stock inc
party government minister election vote	party government election minister opposition

Tabla 5.4: Ejemplo de tópicos hallados para Reuters

Capítulo 6

Conclusiones

En este trabajo se propuso realizar un preprocesamiento a un corpus para agruparlo primero en cubetas que posiblemente están compuestas en su mayoría de un mismo tópico. Posteriormente se utilizaron estas cubetas y una modificación al algoritmo estándar CGS para realizar una actualización en bloque usando cada cubeta.

La hipótesis del trabajo consistió en que los conjuntos formados por Min-Hashing podían ser integrados en el proceso de muestreo del CGS de manera que se realizaran actualizaciones en bloque. El algoritmo presentado logra dicha actualización y los resultados muestran que ambos algoritmos convergen a la misma solución.

El algoritmo propuesto alcanza niveles más altos de log verosimilitud por iteración comparado contra el CGS estándar. Se encontró que a mayor tamaño de tupla la diferencia se ve disminuida, esto debido a que mientras más grande el tamaño de tupla las cubetas tienen menor tamaño y el algoritmo se aproxima al CGS.

Tanto en Reuters como en Twenty Newsgroups se logró estar por encima en log-verosimilitud para diferentes tamaños de tupla y sobre todas las iteraciones. La coherencia evaluada al final de las

iteraciones resultó ser muy similar con ambos métodos y para los diferentes tamaños de tupla evaluados.

Dado que con cubetas grandes se requiere multiplicar muchas probabilidades, se utilizan log-probabilidades y se suman, para finalmente normalizarse. Esto produce un aumento notable en tiempo de ejecución y es un área de mejora importante. Asimismo, como se requiere almacenar las ocurrencias de cada palabra en cada cubeta, el uso de memoria es mayor al del CGS, lo cual representa una desventaja.

Con corpus grandes como el de Reuters se observó que el tiempo para realizar el preprocesamiento aumenta considerablemente, por lo que se optó por particionar horizontalmente la matriz documento-término y obtener las cubetas sobre las particiones para finalmente juntarlas. Sería deseable poder particionar la matriz verticalmente (sobre las palabras) de manera que al juntar las cubetas pudiera haber traslapes; sin embargo, calcular los valores Min-Hash se vuelve costoso conforme aumenta el número de documentos.

El trabajo futuro consiste en encontrar una forma eficiente de calcular las probabilidades de las cubetas y en acelerar el preprocesamiento.

Referencias

Asuncion, A., Welling, M., Smyth, P., y Teh, Y. W. (2009). On smoothing and inference for topic models. En *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence* (p. 27–34). Arlington, Virginia, USA: AUAI Press.

Bezanson, J., Edelman, A., Karpinski, S., y Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. doi: 10.1137/141000671

Blei, D. M. (2012, abril). Probabilistic topic models. *Commun. ACM*, *55*(4), 77–84. Descargado de `https://doi.org/10.1145/2133806.2133826` doi: 10.1145/2133806.2133826

Blei, D. M., Ng, A. Y., y Jordan, M. I. (2003, marzo). Latent dirichlet allocation. *J. Mach. Learn. Res.*, *3*, 993–1022.

Broder, A. Z., Charikar, M., Frieze, A. M., y Mitzenmacher, M. (1998). Min-wise independent permutations (extended abstract). En *Proceedings of the thirtieth annual acm symposium on theory of computing* (p. 327–336). New York, NY, USA: Association for Computing Machinery. Descargado de `https://doi.org/10.1145/276698.276781` doi: 10.1145/276698.276781

Carpenter, B. (2010). Integrating out multinomial parameters in

latent dirichlet allocation and naive bayes for collapsed gibbs sampling..

Foulds, J., Boyles, L., DuBois, C., Smyth, P., y Welling, M. (2013). Stochastic collapsed variational bayesian inference for latent dirichlet allocation. En *Proceedings of the 19th acm sigkdd international conference on knowledge discovery and data mining* (p. 446–454). New York, NY, USA: Association for Computing Machinery. Descargado de `https://doi.org/10.1145/2487575.2487697` doi: 10.1145/2487575.2487697

Fuentes-Pineda, G., y Meza-Ruiz, I. V. (2019, Dec). Topic discovery in massive text corpora based on min-hashing. *Expert Systems with Applications*, *136*, 62–72. Descargado de `http://dx.doi.org/10.1016/j.eswa.2019.06.024` doi: 10.1016/j.eswa.2019.06.024

Griffiths, T. L., y Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences*, *101*(suppl 1), 5228–5235. Descargado de `https://www.pnas.org/content/101/suppl_1/5228` doi: 10.1073/pnas.0307752101

Hoffman, M. D., Blei, D. M., y Bach, F. (2010). Online learning for latent dirichlet allocation. En *Proceedings of the 23rd international conference on neural information processing systems - volume 1* (p. 856–864). Red Hook, NY, USA: Curran Associates Inc.

Kulis, B., y Grauman, K. (2009). Kernelized locality-sensitive hashing for scalable image search. En *2009 ieee 12th international conference on computer vision* (p. 2130-2137).

Lau, J. H., y Baldwin, T. (2016, junio). The sensitivity of topic coherence evaluation to topic cardinality. En *Proceedings of the 2016 conference of the north American chapter of the association for computational linguistics: Human language technologies* (pp. 483–487). San Diego, California: Association for Computational Linguistics. Descargado

de https://www.aclweb.org/anthology/N16-1057 doi: 10.18653/v1/N16-1057

Liu, L., Tang, L., Dong, W., Yao, S., y Zhou, W. (2016, 09). An overview of topic modeling and its current applications in bioinformatics. *SpringerPlus*, *5*. doi: 10.1186/s40064-016-3252-8

Liu, Z., Zhang, Y., Chang, E. Y., y Sun, M. (2011, mayo). Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing. *ACM Trans. Intell. Syst. Technol.*, *2*(3). Descargado de https://doi.org/10.1145/1961189.1961198 doi: 10.1145/1961189.1961198

Luo, C., y Shrivastava, A. (2019). Scaling-up split-merge mcmc with locality sensitive sampling (lss). En *Aaai*.

Miao, Y., Grefenstette, E., y Blunsom, P. (2017, 06–11 Aug). Discovering discrete latent topics with neural variational inference. En D. Precup y Y. W. Teh (Eds.), (Vol. 70, pp. 2410–2419). International Convention Centre, Sydney, Australia: PMLR. Descargado de http://proceedings.mlr.press/v70/miao17a.html

Miao, Y., Yu, L., y Blunsom, P. (2016). Neural variational inference for text processing. En *Proceedings of the 33rd international conference on international conference on machine learning - volume 48* (p. 1727–1736). JMLR.org.

Mimno, D., Hoffman, M. D., y Blei, D. M. (2012). Sparse stochastic inference for latent dirichlet allocation. En *Proceedings of the 29th international coference on international conference on machine learning* (p. 1515–1522). Madison, WI, USA: Omnipress.

Newman, Lau, J. H., Grieser, K., y Baldwin, T. (2010). Automatic evaluation of topic coherence. En (p. 100–108). USA: Association for Computational Linguistics.

Newman, D., Asuncion, A., Smyth, P., y Welling, M. (2009, diciembre). Distributed algorithms for topic models. *J. Mach. Learn. Res.*, *10*, 1801–1828.

Newman, D. J., y Block, S. (2006). Probabilistic topic decomposition of an eighteenth-century american newspaper. *Journal of the American Society for Information Science and Technology*, *57*(6), 753-767. Descargado de `https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.20342` doi: https://doi.org/10.1002/asi.20342

Rajaraman, A., y Ullman, J. D. (2011). *Mining of massive datasets.* USA: Cambridge University Press.

Rezaee, M., y Ferraro, F. (2020, 10). A discrete variational recurrent topic model without the reparametrization trick..

Russell, B. C., Freeman, W. T., Efros, A. A., Sivic, J., y Zisserman, A. (2006). Using multiple segmentations to discover objects and their extent in image collections. En *2006 ieee computer society conference on computer vision and pattern recognition (cvpr'06)* (Vol. 2, p. 1605-1614). doi: 10.1109/CVPR.2006.326

Shrivastava, A., y Li, P. (2014, 04). Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). *Advances in Neural Information Processing Systems*, *3*.

Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., y Freeman, W. (2005). Discovering object categories in image collections..

Slaney, M., y Casey, M. (2008a). Locality-sensitive hashing for finding nearest neighbors [lecture notes]. *IEEE Signal Processing Magazine*, *25*(2), 128-131.

Slaney, M., y Casey, M. (2008b, marzo). Locality-sensitive hashing for finding nearest neighbors [lecture notes]. *IEEE Signal Processing Magazine*, *25*(2), 128-131. doi: 10.1109/MSP.2007.914237

Spring, R., y Shrivastava, A. (2017, 03). A new unbiased and efficient class of lsh-based samplers and estimators for partition function computation in log-linear models..

Srivastava, A., y Sutton, C. (2017). *Autoencoding variational inference for topic models.*

Teh, Y. W., Newman, D., y Welling, M. (2006). A collapsed variational bayesian inference algorithm for latent dirichlet allocation. En *Proceedings of the 19th international conference on neural information processing systems* (p. 1353–1360). Cambridge, MA, USA: MIT Press.

Wang, X., y Grimson, E. (2008). Spatial latent dirichlet allocation. En J. Platt, D. Koller, Y. Singer, y S. Roweis (Eds.), *Advances in neural information processing systems* (Vol. 20, pp. 1577–1584). Curran Associates, Inc. Descargado de `https://proceedings.neurips.cc/paper/2007/file/ec8956637a99787bd197eacd77acce5e-Paper.pdf`

Yuan, J., Gao, F., Ho, Q., Dai, W., Wei, J., Zheng, X., ... Ma, W.-Y. (2015). Lightlda: Big topic models on modest computer clusters. En *Proceedings of the 24th international conference on world wide web* (p. 1351–1361). Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee. Descargado de `https://doi.org/10.1145/2736277.2741115` doi: 10.1145/2736277.2741115

Zhang, H., Chen, B., Guo, D., y Zhou, M. (2020). *Whai: Weibull hybrid autoencoding inference for deep topic modeling.*

Zhang, X., y Sisson, S. A. (2016). *Blocking collapsed gibbs sampler for latent dirichlet allocation models.*