UNIVERSIDAD SANTO TOMAS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA

Facultad de Estadística
Enero 2022, pp. 1–80

ANÁLISIS DE TEMAS UTILIZANDO TWITTER: UNA APLICACIÓN DEL MODELO LDA AL CASO COLOMBIANO.

Manuel Alejandro Diaz Rubiano[a]
manueldiazr@usantotomas.edu.co

Tutor: William Arley Rincon

Resumen

En la actualidad, los avances tecnológicos han logrado que las personas estén cada vez más conectadas entre sí. Las redes sociales han facilitado la comunicación instantánea no solo entre personas que se conocen ya sea por ser amigos, familiares, pareja, sino también entre personas desconocidas que comparten cosas en común, o nada en común.

De acuerdo con (Alvino, 2021), Twitter es una de las redes sociales más usadas en el mundo, con más de 322 millones de usuarios ha cifras del mes de julio del año 2021, y se basa en la publicación de microblogs en los cuales las personas publican por escrito principalmente, lo que están pensando en el momento. Es una red social perfecta para aquellas personas que, por su reconocimiento o fama, pueden influir fácilmente con sus opiniones, gracias a su gran número de admiradores o seguidores.

Twitter en los últimos años ha diseñado una API la cual permite poder conectarse a la información que cada segundo se produce en su plataforma, de los más de 322 millones de usuarios. Para acceder a esta información, se utilizan 2 lenguajes de programación principales: R y Python.

En el actual trabajo, se utiliza la API de Twitter, pues gracias a esta herramienta, se pueden descargar los datos, las opiniones que se están posteando en tiempo real a cada minuto en la red social. Con estos datos, se pueden realizar análisis, por ejemplo, análisis exploratorios sobre las opiniones de las personas, que piensan de las elecciones locales en las ciudades capitales, pues se pueden aplicar diferentes tipos de filtros de ubicación, y obtener datos de ciudades en específico, lo cual se realiza en este trabajo.

Con lo anterior, también se usará distintas técnicas de Procesamiento del Lenguaje Natural (NLP), para poder obtener conclusiones correspondientes a los temas de interés, y a través de los hallazgos, obtener resultados. La principal técnica que se usó en el siguiente trabajo corresponde a las técnica de modelado de tópicos, en especial, el modelo LDA (*Latent Dirichlet Allocation* por sus siglas en ingles).

[a]Estudiante Universidad Santo Tomás

La asignación de Dirichlet Latente (LDA) es un modelo estadístico generativo que permite explicar conjuntos de observaciones mediante grupos no observados que explican por qué algunas partes de los datos son similares (Blei et.al 2003).

Se puede decir que existen dos clases de modelados de tópicos, que son los modelos de tópicos lineares y los modelos de tópicos probabilísticos. Un ejemplo del modelo lineal es el modelo de Asignación Semántica Latente (LSA) y un ejemplo del modelo probabilístico es el modelo de Asignación de Dirichlet Latente, que es el modelo principal del actual trabajo. Se han desarrollado trabajos similares, el cual uno de ellos es el de Guarnizo, E. & Monroy, A. (2021), en el cual se trato de inferir la aceptabilidad publica sobre el tema de la jurisdicción especial para la paz, utilizando opiniones sobre este tema en específico, tomado de redes sociales.

Estos resultados se pueden implementar en cuestiones de política pública, o de influencia política, teniendo en cuenta el panorama pre electoral que afronta Colombia, y aún más con los sucesos como la pandemia de Covid 19, el Paro Nacional, o el recrudecimiento de la violencia. (Haselmayer Jenny, 2016)

***Palabras clave*:** Unigrama, Bigrama, Stemming, Lematización, Cluster, Algoritmo, Naive Bayes, Lenguaje Python, Twitter, Curva ROC, Matriz de Confusión, Datos no Balanceados..

Abstract

Today, technological advances have made people more and more connected to each other. Social networks have facilitated instant communication not only between people who know each other either by being friends, family, partner, but also between unknown people who share things in common, or nothing in common.

According to (Alvino, 2021), Twitter is one of the most used social networks in the world, with more than 322 million users has figures for the month of July 2021, and is based on the publication of microblogs in which people mainly publish in writing what they are thinking at the moment. It is a perfect social network for those people who, due to their recognition or fame, can easily influence their opinions, thanks to their large number of fans or followers.

Twitter in recent years has designed an API which allows you to connect to the information that is produced every second on its platform, from the more than 322 million users. To access this information, 2 main programming languages are used: R and Python.

In the current work, the Twitter API is used, because thanks to this tool, you can download the data, the opinions that are being posted in real time every minute on the social network. With this data, analyzes can be carried out, for example, exploratory analyzes on people's opinions about local elections in capital cities, since different types of location filters can be applied, and data can be obtained from specific cities, which is done in this job. In addition, the classification will be made between whether what users post has a positive or negative feeling.

With the above, different Natural Language Processing (NLP) techniques will also be used, in order to obtain conclusions corresponding to the topics of interest, and through the findings, obtain results. The main technique that will be used in the following work corresponds to topic modeling techniques, especially the LDA (*Latent Dirichlet Allocation* model).

The Latent Dirichlet Allocation (LDA) is a generative statistical model that allows explaining sets of observations by means of unobserved groups that explain why some parts of the data are similar (Blei et.al 2003).

It can be said that there are two kinds of topic modeling, which are linear topic models and probabilistic topic models. An example of the linear model is the Latent Semantic Assignment (LSA) model and an example of the probabilistic model is the Dirichlet Latent Assignment model, which is the main model of real work. Similar works have been developed, one of which is Guarnizo, E. & Monroy, A. (2021), in which public acceptability on the subject of the special jurisdiction for peace was tried to be inferred, using opinions on this specific topic, taken from social networks.

These results can be implemented in matters of public policy, or political influence, taking into account the pre-electoral panorama that Colombia faces, and even more so with events such as the Covid 19 pandemic, the National Strike, or the resurgence of violence. (Haselmayer Jenny, 2016)

***Keywords*:** N-grams, Stemming, Lemmatization, Cluster, Algorithm, Naive Bayes, Python Language, Twitter, ROC Curve, Confusion Matrix, Unbalanced Data.

Índice

Índice de figuras

Índice de cuadros

1. Introducción

En la actualidad, se tiene un gran abanico de posibilidades en la web, por las cuales puedes expresarte, dar tu opinión, dar tu punto de vista, expresarte. Las redes sociales han sido una de las mas grandes invenciones de comienzos del siglo XXI, y con la ayuda de estas redes sociales, se ha logrado masificar la información de las personas de una forma sin precedentes, y que no se hubiese imaginado en los años anteriores a estos, siempre por el principal medio de comunicación de la humanidad por siglos: la escritura.

Los datos están en todas partes, y es menester el poder analizarlos, ya sea para una empresa, una entidad estatal, un gobierno, o para tu propio beneficio. En la actualidad, que la gran mayoría de las personas tienen redes sociales, ya sean Facebook, Twitter, Instagram, o algunas más actuales como TikTok, buscan una cosa en especial, y es reconocimiento, asi como influenciar a un determinado grupo de personas, lo cual puede tener distintos fines (Mainou, 2019).

En el actual trabajo, se descargará información generada en Twitter, y se analizará. La mayoría de las personas están acostumbradas al análisis de datos ordenados en formatos convencionales, es decir, en tablas, matrices, y demás; en la actualidad, esta tomando gran relevancia el análisis de datos no convencionales, a los cuales pertenece los datos textuales que usaran en el presente trabajo, pero también están datos almacenados en audio, imágenes, video, y demás. Pero, ¿Cómo se analizan datos textuales?

Desde hace varios, se ha venido desarrollando la disciplina de la minería de textos, o procesamiento del lenguaje natural, pero en los últimos años, el gran desarrollo computacional ha logrado potenciar esta disciplina mucho más, logrando procesar cantidades de datos gigantescas, en tiempos reducidos, que es preciso lo que se intenta en el actual trabajo. En especial, las técnicas de modelado de tópicos han irrumpido en la actualidad, y se están utilizando por herramientas como el motor de búsqueda de Google, o la pagina de e commerce de Amazon.

2. Planteamiento del Problema

Colombia ha vivido unos periodos bastante atípicos, agitados por levantamientos sociales, pandemias, y un escenario pre elecciones presidenciales bastante álgido. No ha sido un periodo fácil para el país, y esto ha resentido a gran parte de la población colombiana.

Según el Portal Valora Analitik, el desempleo subió a niveles del 20 % durante el año 2020, y comenzó a tener una corrección en el año 2021 , pero muy leve, para que después de unos meses, hubiese estallado el paro nacional, el cual también causo graves estragos en cuanto a la economía del país. También la gran percepción de corrupción que afronta el gobierno actual y su gran impopularidad han sido combustible para las protestas sociales más grandes de las últimas décadas en Colombia (AFP, 2021).

Las redes sociales se han venido desarrollando desde finales de la segunda guerra mundial (al existir la necesidad de crear un medio de comunicación sin la necesidad de implementar conexiones físicos, sino inalámbricas e invisibles a los ojos de los enemigos), pero con el gran avance del internet en los últimos años, y el desarrollo de dispositivos de comunicación más pequeños, más accesibles para el público en general, y mucho más fáciles de llevar consigo mismos, han creado un nicho de comunicación instantánea, llegando a lo

que en la actualidad se tienen como las redes sociales.

En específico, se tiene principal consideración de la Red Social Twitter. Esta red social nació en el año 2006 en San Francisco, California, EE.UU. El principal impulsor y creador es Jack Dorsey.

En la actualidad Twitter ostenta una red de usuarios de más de 300 millones a nivel mundial, aproximadamente unos 322 millones a nivel mundial que están activos; en Colombia esta red social tiene un estimado de más de 3 millones de usuarios diarios, entre personas, empresas, entidades públicas, fundaciones, universidades y demás, que publican contenido a cada segundo y a diario (Ahlgren, 2021).

Mediante lo planteado con anterioridad, la pregunta de investigación formulada es la siguiente: ¿Es posible identificar y clasificar por temas, lo que los colombianos están expresando textualmente en Twitter, mediante técnicas de NLP (Natural Language Processing) y Modelado de Tópicos?.

2.1. Justificación

Es muy importante conocer los ánimos de los usuarios de Twitter por medio de un análisis de los tweets generados en las siguientes ciudades colombianas: Bogotá, Pereira, Medellín, Cúcuta, Cartagena y Cali, entre los meses de junio a octubre del 2021, ya que en estas ciudades vive la gran mayoría de la población colombiana.

Esto permitiría identificar los ánimos de los usuarios de Twitter al catalogar sus publicaciones entre positivo o negativo, así como también poder conocer las temáticas que se están tratando en esta red social; a través de lo anterior, poder tener un panorama general de las principales temáticas tratadas por los usuarios sin la necesidad de gastar valiosos recursos monetarios y de tiempo, en el diseño y realización de una encuesta. Con este trabajo, se puede diseñar un esquema de política pública, de campaña política, o decisiones de mejoramiento de imagen pública, que favorezcan la favorabilidad y la aceptación del público.

3. Objetivos

A continuación, se presentan los objetivos específicos y el objetivo general.

Objetivo general

Analizar los textos de las publicaciones de los usuarios de Twitter, mediante técnicas estadísticas de modelado de tópicos, minería de textos, con enfoque principalmente en los temas más álgidos en la actualidad del contexto colombiano.

Objetivos específicos

- Construir una base de datos con respecto a los temas políticos más álgidos en pleno año pre electoral, utilizando la API que ofrece Twitter.
- Realizar análisis exploratorio esencial de los datos recopilados.

- Entrenar un modelo clasificatorio utilizando como etiqueta la polaridad de los datos.
- Implementar un modelo estadístico de Topic Modeling, que permita realizar la clasificación.
- Implementar el modelo clasificatorio entrenado previamente para clasificar los tweets recopilados mediante la API de Twitter.
- Evaluar el rendimiento de los modelos clasificatorios utilizando métricas como F1-score, Curva ROC y métricas de coherencia de tópicos.

4. Hipótesis

Por medio de procesos y métodos estadísticos, es posible analizar masivamente información textual generada en las redes sociales, específicamente Twitter, y obtener conclusiones sobre lo que las personas están hablando y opinando en su entorno social, y poder clasificar posteriormente entre positivo o negativo.

5. Marco Referencial

En las siguientes secciones se expondrán los lineamientos necesarios para el actual proyecto.

5.1. Marco Teórico

Se tienen como principales métodos, los métodos bayesianos, y de regresión.

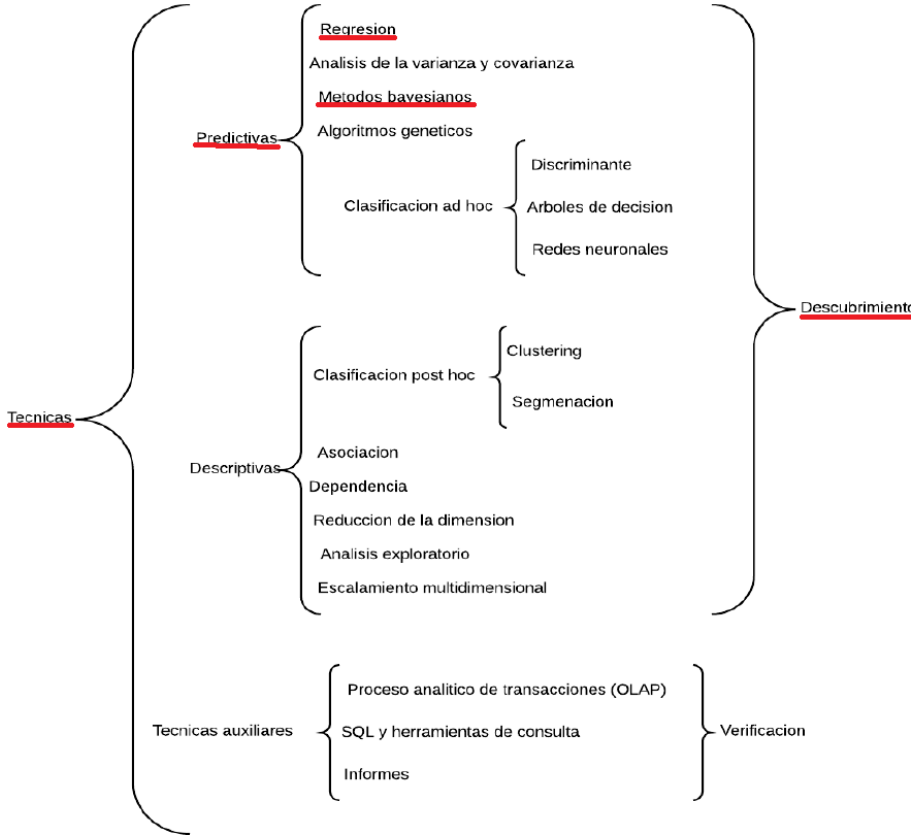


Figura 1: Fuentes de datos
Fuente:https://repository.ucatolica.edu.co/bitstream/10983/24981/1/JEPDocumentoFinal.pdf

5.2. Marco Conceptual

NOTA: Las fuentes de las definiciones se encuentran en el pie de página.

Procesamiento del Lenguaje Natural: es una rama derivada de la Inteligencia Artificial, que se encarga de investigar, facilitar y desarrollar métodos que faciliten la comuni-

cación entre personas y maquinas, mediante los lenguajes convencionales de las personas como ingles o español.

Minería de Datos: es un proceso especial mediante el cual se identifican patrones en los datos, asi como hallar anomalías y relaciones ente los mismos datos. Se puede aplicar tanto a grandes volúmenes de datos, como en pocos datos implementando métodos como estadística no paramétrica. Con el análisis de datos, se puede predecir y clasificar para la correcta toma de decisiones de alguna organización o negocio.

Análisis de Sentimientos: es el proceso mediante el cual se detecta el tono emocional (positivo, negativo, neutro) y además se pueden detectar sentimientos de ira, amor, compasión y demás, con el fin de obtener información relevante con respecto a los clientes tratándose de una empresa, o ciudadanos tratándose de un gobierno que sea el que este interesado. [1]

Estadística Frecuentista: es el tipo de estadística mas conocida y mas antigua, y se basa en los conceptos básicos y más clásicos de la probabilidad, y también en el contraste de hipótesis.

Estadística Bayesiana: una de las grandes diferencias entre la estadística frecuentista y la estadística bayesiana, es que la estadística bayesiana integra información exógena, o externa, al estudio que se este realizando en la actualidad, por lo cual la probabilidad de un determinado suceso, se ve afectada por la información de un suceso previo que ya ocurrió. [2]

Aprendizaje Supervisado: los algoritmos o modelos, que se basan en una forma de aprendizaje supervisado, son aquellos que basan su aprendizaje en un conjunto de datos previamente etiquetados. Existen algoritmos y modelos de aprendizaje supervisado tanto para datos con atributos numéricos, como para datos con atributos categóricos.

Aprendizaje No Supervisado: a diferencia de aprendizaje supervisado, aquí el aprendizaje se basa en conjuntos de datos sin ninguna clase de etiqueta. Esta clase de algoritmos están enfocados principalmente en tareas de agrupamiento o también llamadas segmentación. Los principales métodos de agrupamiento son los métodos jerárquicos, y los métodos no jerárquicos o particionales.[3]

[1] Haselmayer, M., Jenny, M. (2016)
[2] (van Zyl , 2018)
[3] (Besancon Rajman, 2013)

6. Antecedentes

En los siguientes apartados se presentaran los antecedentes directos, y unos de los mas relevantes para la realización del presente trabajo.

6.1. Teorema de Bayes

El teorema de bayes es uno de los teoremas más usados en la estadística. Este teorema describe la probabilidad de ocurrencia de un determinado evento dado que un evento relacionado ya ocurrió y se tiene conocimiento de ello. Pero antes del teorema de bayes, es necesario tener en consideración el teorema de probabilidad total.[4]

En pocas palabras, el teorema de probabilidad total permite calcular la probabilidad de un suceso a partir de probabilidades condicionales. [5] Sea $X_1, X_2, \ldots, X_n$ un conjunto del espacio muestral y sea Y un evento cualquiera que sea y que se conocen las probabilidades condicionales entonces la probabilidad del suceso Y viene dada por la siguiente expresión: [6]

$$P(Y) = P(Y|X_1)P(X_1) + P(Y|X_2)P(X_2) + \cdots + P(Y|X_n)P(X_n) \tag{1}$$

$$P(Y) = \Sigma_{i=1}^{n} P(Y|X_i)P(X_i) \tag{2}$$

Ahora, si decimos que la probabilidad de ocurrencia de cualquier evento Y es $P(Y) > 0$, nos conlleva a la ecuación 1 vista anteriormente.

6.2. Minería de Textos

Según Bengfort, B., Bilbro, R. Ojeda, T. (2018), la minería de textos, o text Mining en ingles, son aquellas técnicas desarrolladas a partir de la minería de datos elemental, aplicada al proceso de arreglo, procesamiento, extracción de resultados y presentación de los mismos, a partir de los datos obtenidos. Estos datos pueden ser obtenidos de diferentes maneras, las cuales no se detallarán en el presente documento. La gran diferencia del análisis de datos textuales a través de la minería de textos, contra la minería de datos normales, es la calidad y forma en que los datos están contenidos, y por decirlo de una manera, el etiquetado de los datos. Mientras que, en la minería de datos clásica, los datos tienden a estar en una forma lista para realizar cualquier consulta, o como se ha tornado en termino famoso, datos tipo SQL, (debido al lenguaje de programación SQL) los cuales están eventualmente organizados y etiquetados en columnas y filas, los datos textuales vienen en forma NoSQL, o se podría decir de forma desorganizada, en su forma natural (Dave Sharma, 2012).

[4](Hart Duda, 1973)
[5](Lin, 2016)
[6](Kontostathis, 2004)

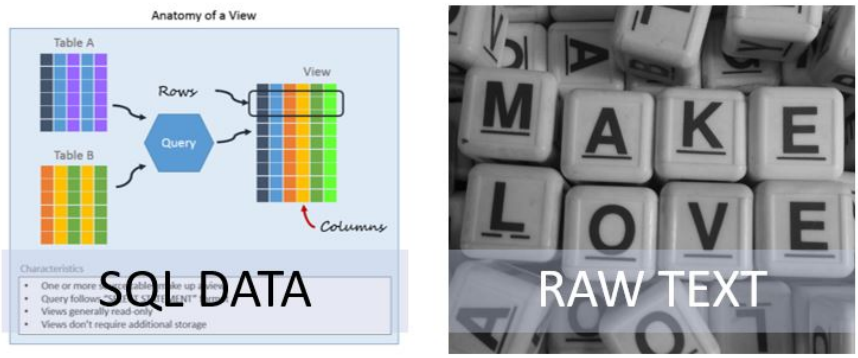


Figura 2: Tipos de datos
Fuente:
https://www.datamation.com/big-data/structured-vs-unstructured-data/

Como se puede notar en la ilustración 1, existen prácticamente dos tipos de datos; entre ellos se encuentran los datos organizados de una forma bastante entendible para el lenguaje de máquina, que en su gran mayoría, deberían ser bases relacionales, con sus key index excelentemente identificados, o en unas pocas palabras, tablas de datos con un índice que las identifica y las relacione con otras bases.

Pero en la realidad, la gran mayoría de los datos que se producen en tiempo real no están en esta forma. los datos producidos por los seres humanos, están casi siempre en sonidos, imágenes, medios audiovisuales, obras de arte, pinturas, o en nuestro caso, en texto.

La técnica de minería de textos trata de volver estos datos textuales que se generan a cada segundo, por miles de millones de líneas de textos en los miles de plataformas web, en datos que sean entendibles para el lenguaje de máquina.

Para que estos datos sean entendibles por el lenguaje de máquina, se debe primero filtrar estos datos y ser tratados, en los cual se describen procesos como pones las letras en minúsculas, quitar letras seguidas bastantes repetidas, y demás procesos que se realizan y se explicaran más a fondo posteriormente. Algunas de las técnicas principales aplicadas al text Mining son las siguientes:

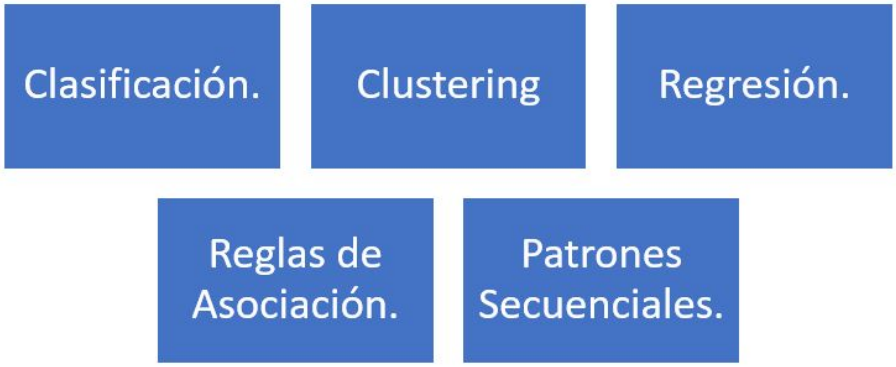


Figura 3: Flujo de text mining
Fuente: elaboración propia

6.3. Modelado de Tópicos

El modelado de temas es parte de una clase especial de métodos de análisis de textos (Vázquez Marcos, 2017). Estos analizan 'bolsas' o grupos de palabras juntas, o corpus de palabras, en lugar de contarlas individualmente, para capturar cómo el significado de las palabras depende del contexto más amplio en el que se usan en términos naturales. El modelado de tópicos no es el único método que hace esto: el análisis de conglomerados que es uno ampliamente usado por su rapidez, la Asignación de Dirichlet Latente, que es uno de los temas principales de este documento, y que se analizara con más detalle más adelante, y otras técnicas también se han utilizado para identificar los corpus dentro de los textos. El modelado de tópicos tiene dos ventajas que lo hacen resaltar sobre las formas simples de análisis de conglomerados de palabras, o bolsas de palabras, como lo realizan algoritmos como k-medias: en las agrupaciones como las de k-medias cada observación se debe asignar a tan solo una agrupación de las existentes, por el contrario el modelado de tópicos a cada palabra se le asigna una probabilidad de pertenecer a algún tópico, lo cual da la flexibilidad de poder pertenecer a varios tópicos, pero con mayor representatividad en uno.

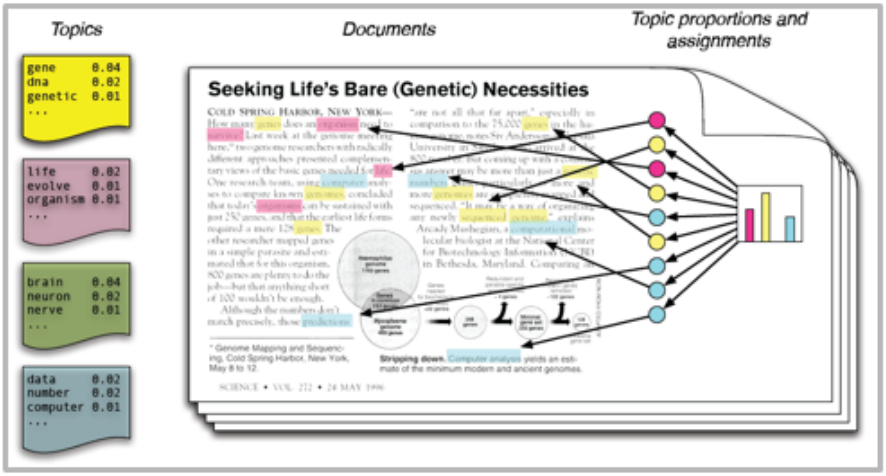


Figura 4: Modelado de tópicos
Fuente:
https://www.analyticsvidhya.com/wp-content/uploads/2016/08/Modeling1.png

6.4. Clasificador Bayes Ingenuo

Naive bayes, o como se le llama a veces en español, el Bayes Ingenuo, es uno de los algoritmos clasificatorios más influyentes y más usados para el proceso de clasificación, valga la redundancia. Este algoritmo está basado en el teorema de Thomas Bayes (matemático inglés y ministro presbiteriano), que supone la independencia de los predictores. Los anterior significa la asunción de que el efecto de una característica particular en una clase es independiente de otras características. Esta asunción es la que se considera ingenua (y de ahí el origen del nombre del algoritmo) y además simplifica la computación (Hart Duda, 1973).

$$P(x|Y) = \frac{P(Y|x)P(x)}{P(Y)} \tag{3}$$

Donde:

- $p(x)$ =probabilidad de que x sea cierta que hace referencia a la hipotesis.
- $P(Y)$ = probabilidad de que 'Y' sea cierta, y suele hacer referencia a los conjuntos de datos.
- $P(x|Y)$ = probabilidad de 'x' dado que 'Y' es cierta.
- $P(Y|x)$ = probabilidad de 'Y' dado que 'x' es cierta.

Una de las principales ventajas del algoritmo de Bayes Ingenuo es, que, si en los casos de datos las variables tienen independencia total, el algoritmo necesita muchos menos datos de entrenamiento, y su desempeño es notablemente superior al de otros algoritmos clasificatorios como lo es la regresión logística o los árboles de decisión. La desventaja

principal de este algoritmo viene de la asunción de independencia de las variables, puesto que esto en la vida real, muchas veces es casi imposible.

6.5. Indexación Semántica Latente (LSI).

La indexación semántica latente o LSI por sus siglas en inglés, es un método de procesamiento de lenguaje natural, y de análisis de tópicos. Su finalidad es analizar las relaciones entre un set de documentos y sus tópicos. Este método es el precursor del modelo Latent Dirichlet Allocation (LDA por sus siglas en ingles).[7]

El modelo de Indexación Semántica latente asume que las palabras se repiten en segmentos de documentos con temáticas similares entre sí. En este algoritmo se utiliza la técnica de descomposición por valores singulares, en una matriz que se debe diseñar también que contiene el recuento por ocurrencia de cada palabra, para reducir el numero de columnas y conservando la estructura de similitud entre columnas; el anterior procedimiento genera una estructura de vector por cada documento o tópico. Los tópicos se comparan posteriormente tomando el producto escalar entre la forma normalizada de dos vectores, formado por dos columnas aleatorias. los valores cercanos a uno representan que dos documentos son totalmente similares entre sí, y valores cercanos a cero representa que dos documentos son totalmente diferentes entre sí.

Aunque el modelo de Indexación semántica latente suena muy similar al modelo de Asignación de Dirichlet Latente, tienen sus diferencias entre sí, aunque el propósito de las dos técnicas es la misma. La indexación semántica latente, se entrena mediante el uso de el uso de una matriz de descomposición, o método de descomposición por valores singulares, en la matriz de frecuencia de palabras, mientras que el modelo de asignación de Dirichlet Latente es un modelo generativo bayesiano, que asume una probabilidad de Dirichlet a priori sobre los tópicos latentes.

En la práctica, el modelo LSI es mucho más rápido de entrenar sobre los datos, lo cual puede ser una gran ventaja cuando se tiene un set de datos de inmensas dimensiones, pero la precisión y el performance del modelo LDA es mucho mejor, lo cual es una gran ventaja cuando se quiere conocer en especial la polaridad de sentimientos de determinados textos.[8]

[7](Kontostathis, 2004)
[8](Lin, 2016)

7. Asignación de Dirichlet Latente.

Para el estudio de conocimiento acumulado con respecto a la temática de este trabajo, se consideraron como esenciales los temas a continuación.

7.1. Introducción al Modelo LDA.

La Asignación de Dirichlet Latente, es uno de los algoritmos generativos más usados y con mejor desempeño en cuanto a modelado de tópicos. El algoritmo fue propuesto originalmente en el contexto de poblaciones genéticas, por J.K. Pritchard, M. Stephens y P.Donelly (Pritchard JK, 2000), pero posteriormente fue aplicado al contexto de aprendizaje de máquinas y modelado de tópicos por los docentes de la universidad de California Berkeley, David Blei, Michael I. Jordan, y por el docente de la universidad de Stanford Andrew Ng (Blei et.al 2003). Este algoritmo tiene sus bases probabilísticas en la estadística bayesiana e inferencia estadística bayesiana, pero que gracias a su gran flexibilidad, es fácilmente implementable en lenguaje de maquina y al igual que el algoritmo de Bayes Ingenuo, su asunción de independencia facilita la computación. Otra de las grandes ventajas de este algoritmo, es que se aprovecha de mejor manera los datos que se poseen, sin necesidad de tener que recolectar grandes cantidades de información, gracias a su fundamento bayesiano (Blei, Ng, Jordan, 2003).

En la Asignación de Dirichlet Latente, se considera que cada documento, o bolsa de palabras, tiene una mezcla de varios temas que son asignados a través del mismo algoritmo; en esta parte se supone que la distribución de cada tema proviene de una distribución de probabilidad de Dirichlet, que se explicara con más detalle más adelante en este documento. Esto significa que el algoritmo permite que cada bolsa de palabras pertenezca a diferentes tópicos al mismo tiempo, con un peso diferente para cada tópico; esto quiere decir, cada bolsa de palabras tiene más probabilidad de pertenecer a un determinado tópico que a otro tópico.

Un modelo de LDA puede tener varios temas distintos, como política o deportes. El tema de política puede generar palabras como presidente, alcalde, reforma, contrato, ley, etc., mientras que el tema deportes, puede generar palabras como jugador, futbol, baloncesto, equipo, contrato. Ambos temas pueden generar palabras en común, pero cada bolsa de palabras según las palabras que contenga, tendrá una probabilidad mayor o menor de pertenecer a alguno de los tópicos. Las palabras sin relevancia como “la”, “que”, “como”, tendrán una probabilidad similar entre los tópicos, pero en el pre-procesamiento de los textos se prefiere retirar este tipo de palabras.

7.2. Distribución de Dirichlet

El modelo de LDA establece como hiper parámetros a priori, a α y β, los cuales son muestras de las características de los documentos y tópicos tomados de una distribución de Dirichlet. La distribución de Dirichlet, en esencia, es una generalización multivariante de la distribución Beta, por lo cual a veces se encuentra en la literatura como distribución beta multivariante, con una función de densidad de probabilidad que es la siguiente:

$$\frac{1}{B(\alpha)}\prod_{i=1}^{n} x_i^{\alpha_i - 1} \qquad \text{donde } B(\alpha) = \frac{\prod_{i=1}^{n}\Gamma(\alpha_i)}{\Gamma(\Sigma_{i=1}^{n}\alpha_i)} \tag{4}$$

- α es un vector de elementos positivos.
- Γ es la función gamma con la siguiente forma:

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt$$

- La distribución de Dirichlet es conjugada a priori de la distribución multinomial.

En la siguiente imagen se puede observar ejemplos de la distribución de Dirichlet:

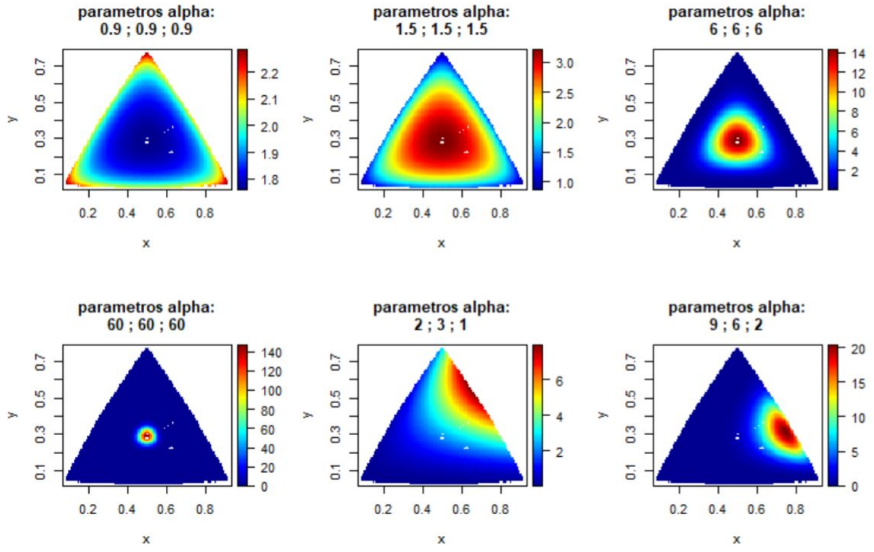


Figura 5: Distribución dirichlet
Fuente: Elaboración propia

7.3. Proceso generativo

Se comienza dando una probabilidad de tema para cada palabra del documento, β. Este valor es elegido aleatoriamente de una distribución de probabilidad de Dirichlet, con parámetro η, dando de esta manera el resultado de la expresión $P(\beta|\eta)$. El paso a seguir es generar documentos, para lo cual se inicia calculando los diferentes tópicos de todos los documentos o bolsas de palabras, θ_d, generado a través de una distribución de probabilidad de Dirichlet con parámetro α, el cual es el resultado al termino $P(\theta|\alpha)$. El tópico $z_{d,n}$, es escogido a partir de una distribución de probabilidad multinomial, con parametro θ. Después de obtener este ultimo resultado, se utiliza la distribución de palabras en $z_{d,n}$, β_k, para determinar la palabra resultante, que es el único resultado que se puede observar, ya que el resto son latentes. Esto es para cada iteración.

1. Para $k = 1$ hasta K, donde K es la suma total de los tópicos.

   1.1 Escoger variables al azar para la distribución de palabras de cada tópico.

   1.2 $\beta_k \sim Dir(\eta)$

2. Para $d = 1$ hasta D, donde D es el número total de documentos.

   2.1 Escoger variables al azar para la distribución de típicos de cada documento.

   2.2 $\theta_k \sim Dir(\alpha)$

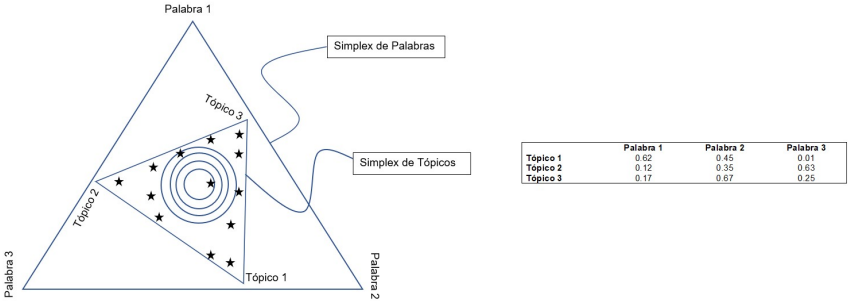
   2.3 Para $w = 1$ hasta W, donde W es el número de palabras en el documento d.

      2.3.1 Escoger el tópico para la palabra w.

      2.3.2 $z_i \sim Multi(\theta_d)$

      2.3.3 Escoger palabra basado en según la distribución de la palabra en el tópico z_i.

      2.3.4 $w_i \sim Multi(\beta_{z,i})$



	Palabra 1	Palabra 2	Palabra 3
Tópico 1	0.62	0.45	0.01
Tópico 2	0.12	0.35	0.63
Tópico 3	0.17	0.67	0.25

Figura 6: Distribución dirichlet - LDA
Fuente: Adaptado de Blei et al., 2003

Como parte del modelo, se tiene la siguiente distribución conjunta:

$$P(w, z, \theta, \beta|\alpha, \eta) = P(\theta|\alpha)P(\beta|\eta)P(z|\theta)P(w|z, \beta)$$

$$= \prod_{d=1}^{D} P(\theta_d|\alpha) \cdot \prod_{k=1}^{K} P(\beta_k|\eta) \cdot \prod_{n=1}^{N}\prod_{d=1}^{D} P(z_{d,n}\theta_d) \cdot \prod_{n=1}^{N}\prod_{d=1}^{D} P(W_{d,n}|z_{d,n}, \beta)$$

En esta ecuación, los dos primeros términos provienen de probabilidades de Dirichlet, y los dos últimos Multinomial.

Como traducción directa de la distribución conjunta, se tiene el modelo gráfico o notación por platos:

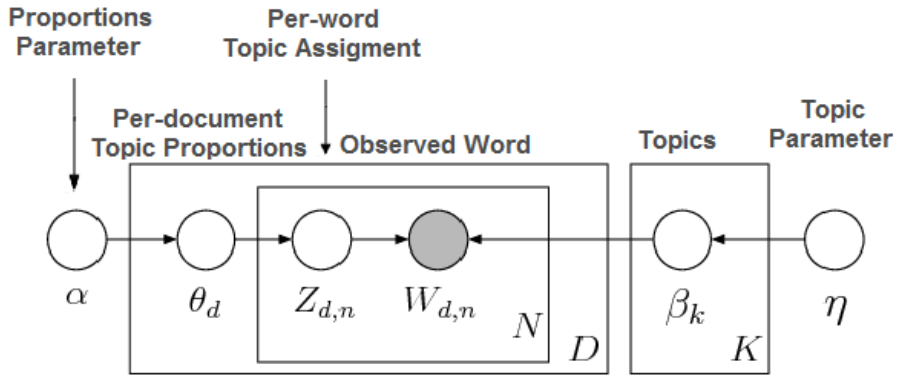


Figura 7: Flujo de modelado de tópicos
Fuente: https://filebox.ece.vt.edu/ s14ece6504/projects/alfadda$_t opic/index.html$

- D: número de documentos o bolsa de palabras.
- N: número de palabras en un documento.
- V: número de palabras en el corpus.
- n: índice auxiliar sobre longitud.
- v: índice auxiliar sobre corpus.
- d: índice auxiliar sobre documentos.
- K: número de tópicos.
- θ : distribución del tema para cada documento.
- α: es el parámetro con distribución de Dirichlet, antes de la distribución de tópicos de los documentos.
- β: distribución de la palabra para cada tópico.
- η: Parámetro anterior con distribución Dirichlet para cada palabra perteneciente a un tópico.

Los modelos gráficos son un método estándar de representar estadísticamente, modelos complejos que están compuestos por varias variables aleatorias, interconectadas en formas complejas. Las 4 probabilidades principales del lado derecho de la ecuación, se pueden entender de la siguiente manera:

- El parámetro $P(\theta|\alpha)$ hace referencia a la distribución de probabilidad por cada documento, que a su vez proviene de una distribución de Dirichlet con parámetro α. Esta probabilidad Dirichlet, con parámetro α es un hiperparametro, que controla la distribución de los tópicos por cada documento. Tiene una forma muy similar a la distribución de Dirichlet, que es la siguiente:

$$P(\theta|\alpha) = \frac{\Gamma(\Sigma_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\prod_{k=1}^{K}\theta_k^{\alpha_k-1}$$

	Topico 1	**Topico 2**	**Topico 3**	**Topico 4**
Documento 1 $\theta_{d=1}$	0.54	0.33	0.05	0.01
Documento 2 $\theta_{d=2}$	0.00	0.75	0.2	0.03
Documento 3 $\theta_{d=3}$	0.85	0.1	0.0	0.0

Como se puede observar en la tabla anterior (valores de ejemplo, no son reales), se tienen las probabilidades de que cada documento albergue diferentes tópicos, teniendo en cuenta de que uno de los principales fundamentos del modelo, es que los documentos están compuestos por múltiples tópicos.

- La distribución de las palabras por tópicos para el corpus, que hace referencia al vocabulario que se define, proviene de una distribución de Dirichlet con parámetro η. La expresión $\beta_{k,v}$ da la probabilidad de que una palabra sea escogida cuando el tópico ya este dado.

$$P(\beta|\eta) = \prod_{k=1}^{K} \frac{\Gamma(\Sigma_{k=1}^{K}\eta_k)}{\prod_{v=1}^{V}\Gamma(\eta_{k,v})} \prod_{v=1}^{V} \beta_{k,v}^{n_{k,v}-1}$$

	Word 1	**Word 2**	**Word 3**	**Word 4**	**Word 5**	**Word 6**
Top 1 $\beta_{k=1}$	0.12	0.1	0.0	0.07	0.15	0.2
Top 2 $\beta_{k=2}$	0.23	0.36	0.1	0.05	0.0	0.0
Top 3 $\beta_{k=3}$	0.0	0.01	0.4	0.12	0.7	0.7
Top 4 $\beta_{k=4}$	0.0	0.0	0.33	0.18	0.15	0.22

- Para la probabilidad de $P(z|\theta)$ que hace parte también de la distribución conjunta, se le encarga la asignación de tópicos hacia las palabras en el corpus, que lleva como indicativo z. Recordar este resultado depende de una distribución de probabilidad multinomial; en la distribución conjunta, la probabilidad de z se expresa como $\eta_{d,k}$, que es el conteo para todos los tópicos y documentos, y que a su vez significa la cantidad de veces que un tópico k ha sido asignado a cualquier palabra en el documento d.

$$P(z|\theta) = \prod_{d=1}^{D}\prod_{k=1}^{K} \theta_{d,k}^{n_{d,k}}$$

	Word 1	**Word 2**	**Word 3**	**Word 4**	**Word 5**	**Word 6**
Doc 1 $z_{d=1,n}$	Top k=1	Top k=2	Top k=2	Top k=4		
Doc 2 $z_{d=2,n}$	Top k=4	Top k=1		Top k=2	Top k=3	
Doc 3 $z_{d=3,n}$	Top k=2		Top k=1	Top k=1	Top k=4	Top k=3

En la tabla anterior, se puede observar que cada palabra, que aparece en un determinado documento, se le asigna un tópico.

- Por último, se tiene la probabilidad del corpus, o del documento que se genera W, dado que se tiene z y β:

$$P(w|z,\beta) = \prod_{k=1}^{K}\prod_{v=1}^{V}\beta_{k,v}^{n_{k,v}}$$

Donde $n_{k,v}$ es la cantidad de veces que un tópico k fue asignado a una palabra v en todo el corpus o documento generado.

	Word 1	**Word 2**	**Word 3**	**Word 4**
Tópico 1	0.53		0.4	0.15
Tópico 2	0.7	0.12	0.36	0.12
Tópico 3	0.63	0.42	0.12	

Como se observa en la tabla anterior, el algoritmo genera un corpus o documento, en el cual se observan las palabras y la probabilidad de que determinada palabra pertenezca a ese tópico.

El problema central para el modelo LDA, es el de determinar la distribución a posteriori de las variables latentes dado el documento.

$$P(z,\theta,\beta|w) = \frac{P(z,\theta,\beta,w)}{\int_z\int_\theta\int_\beta P(z,\theta,\beta,w)d\beta d\theta dz} \tag{5}$$

$$= \frac{P(w|z,\theta,\beta)P(z,\theta,\beta)}{\int_z\int_\theta\int_\beta P(z,\theta,\beta,w)d\beta d\theta dz}$$

Teniendo en cuenta el modelo gráfico, se puede escribir también de la siguiente manera:

$$P(z,\theta,\beta,w) = \Big(\prod_{d=1}^{D}P(\theta_d|\alpha_d)\Big)\Big(\prod_{d=1}^{D}\prod_{n=1}^{N}P(z_{d,n}|\theta_d)\Big) \tag{6}$$

$$\Big(\prod_{d=1}^{D}\prod_{n=1}^{N}P(w_{d,n}|\theta_{d,n},\beta)\Big)\Big(\prod_{k=1}^{K}P(\theta_k|\beta_k)\Big)$$

$$= \Big(\prod_{d=1}^{D}\mathcal{D}(\theta_d,\alpha_d)\Big)\Big(\prod_{d=1}^{D}\prod_{n=1}^{N}(\prod_{k=1}^{K}\theta_{d,k}^{z_{d,n,k}})\Big)\Big(\prod_{d=1}^{D}\prod_{n=1}^{N}(\prod_{k=1}^{K}\beta_{kw_{d,n}}^{z_{d,n,k}})\Big)\Big(\prod_{k=1}^{K}\mathcal{D}(\beta_k,\eta_k)\Big) \tag{7}$$

Se tiene la siguiente propiedad para el producto de:

$$\begin{aligned} P(\theta|\alpha)P(z|\theta) =& \Big(\prod_{d=1}^{D}\Big(\frac{\Gamma(\sum_{k=1}^{K}\alpha_{d,k})}{\prod_{k=1}^{K}\Gamma(\alpha_{d,k})}\cdot\prod_{k=1}^{K}\theta_{d,k}^{\alpha_{d,k}-1}\Big)\Big)\cdot\Big(\prod_{d=1}^{D}\prod_{n=1}^{N}(\prod_{k=1}^{K}\theta_{d,k}^{z_{d,n,k}})\Big) \\ =& \Big(\prod_{d=1}^{D}\frac{\Gamma(\sum_{k=1}^{K})\alpha_{d,k}}{\prod_{k=1}^{K}\Gamma(\alpha_{d,k})}\cdot\prod_{k=1}^{K}\theta_{d,k}^{\alpha_{d,k}-1+n_{d,k}}\Big) \end{aligned} \tag{8}$$

Con el resultado de la propiedad de la ecuación 8, se puede generalizar para la ecuación 7, obteniendo lo siguiente:

$$\Big(\prod_{d=1}^{D}\frac{\Gamma(\Sigma_{k=1}^{K}\alpha_{d,k})}{\prod_{k=1}^{K}\Gamma(\alpha_{d,k})}\prod_{k=1}^{K}\theta_{d,k}^{\alpha_{d,k}-1+n_{d,k}}\Big)\Big(\prod_{k=1}^{K}\frac{\Gamma(\Sigma_{v=1}^{V}\eta_{k,v})}{\prod_{v=1}^{V}\Gamma(\eta_{k,v})}\prod_{v=1}^{V}\beta_{k,v}^{\eta_{k,v}-1+n_{k,v}}\Big) \tag{9}$$

donde: $n_{d,k} := [n_{d,k1}, ..., n_{d,k,V}] \rightarrow n_{d,k,\cdot} = \sum_V n_{d,k,V}$

La anterior expresión es la marginalización sobre las variables latentes. Esto permite obtener la probabilidad de un documento "w", cuando los parámetros α y η están dados.

Se realiza la suma sobre todas las expresiones de z y se integra con respecto a θ y β.

$$\begin{aligned} P(w|\alpha,\eta) = \int_\beta\int_\theta \Sigma_z\Big(\prod_{d=1}^{D}\frac{\Gamma(\Sigma_{k=1}^{K}\alpha_{d,k})}{\prod_{k=1}^{K}\Gamma(\alpha_{d,k})}\prod_{k=1}^{K}\theta_{d,k}^{\alpha_{d,k}-1+n_{d,k}}\Big)\times \\ \Big(\prod_{K=1}^{k}\frac{\Gamma(\Sigma_{v=1}^{V}\eta_{k,v})}{\prod_{v=1}^{V}\Gamma(\eta_{k,v})}\prod_{v=1}^{V}\beta_{k,v}^{\alpha_{k,v}-1+n_{k,v}}\Big)d\theta d\beta \end{aligned} \tag{10}$$

La expresión anterior hace que no se tenga una solución por métodos convencionales. La dupla conformada por las expresiones θ y β, que al calcular *log* de esta función, se hace imposible poder separarlos.

Como no se puede obtener la inferencia exacta, se pueden utilizar varias técnicas para aproximar esta inferencia.

7.4. Estimación del modelo

Varios métodos se han utilizado para la estimación de los parámetros en modelos estadísticos generativos. Para el modelo de Asignación de Dirichlet Latente, existen en la actualidad técnicas de estimación que se basan en métodos de re muestreo basados en MCMC (markov Chain Monte Carlo), y métodos de optimización, que se suelen llamar métodos bayesianos variacionales. (Speh , Muhic , and Rupnik, 2021)

Muestreo de Gibbs colapsado:

En este método, se integra primero sobre los parámetros β y θ (Speh , Muhic , and Rupnik, 2021)

$$P(z,w|\alpha,\eta) = \int_\theta\int_\beta P(z,w,\theta,\beta|\alpha,\eta)d\beta d\theta \tag{11}$$

Por claridad, se observa la ecuación para $P(z,w|\alpha,\eta)$ con θ y β ya integradas

$$
\begin{aligned}
P(z,w|\alpha,\eta) &= \prod_{d=1}^{D} \frac{\Gamma(\Sigma_{k=1}^{K}\alpha_{d,k})}{\prod_{k=1}^{K}\Gamma(\alpha_{d,k})} \cdot \prod_{k=1}^{K} \theta_{d,k}^{\alpha_{d,k}-1+n_{d,k}} \times \\
&\prod_{k=1}^{K} \frac{\Gamma(\Sigma_{v=1}^{V}\eta_{k,v})}{\prod_{v=1}^{V}\Gamma(\eta_{k,v})} \prod_{v=1}^{V} \beta_{k,v}^{\eta_{k,v}-1+n_{k,v}} \\
&= \prod_{d=1}^{D} \frac{\Gamma(\Sigma_{k=1}^{K}\alpha_{d,k})}{\prod_{k=1}^{K}\Gamma(\alpha_{d,k})} \frac{\prod_{k=1}^{K}\Gamma(n_{d,k}+\alpha_{d,k})}{\Gamma(\Sigma_{k=1}^{K}n_{d,k}+\alpha_{d,k})} \times \\
&\prod_{k=1}^{K} \frac{\Gamma(\Sigma_{v=1}^{V}\eta_{k,v})}{\prod_{v=1}^{V}\Gamma(\eta_{k,v})} \frac{\prod_{v=1}^{V}\Gamma(n_{k,v}+\eta_{k,v})}{\Gamma(\Sigma_{v=1}^{V}n_{k,v}+\eta_{k,v})}
\end{aligned}
\tag{12}
$$

Obsérvese que la probabilidad condicional $P(z|\alpha,\eta)$ no depende de z, por lo cual las ecuaciones pueden ser derivadas directamente de $P(z,w|\alpha,\eta)$ el objetivo del algoritmo es aproximar la distribución $P(z,w|\alpha,\eta)$ (Speh , Muhic , and Rupnik, 2021). Para el algoritmo (aplicado al modelo LDA), se comienza ignorando el producto que se obtiene sobre los documentos, y de los tópicos, ya que estos no dependen en la asignación de tópicos $z_{m,n}$ (Geigle, 2017).

Se tiene un interés especial en la siguiente probabilidad condicional:

$$P(w, z_{m,n} = k, z_{-(m,n)}|\alpha,\eta)$$

Donde $z_{-(m,n)}$ significa que se tiene en consideración todos los parámetros de z, menos $z_{m,n}$ (Ponweiser, 2012).

$$P(w, z_{m,n} = k, z_{-(m,n)}|\alpha,\eta) \propto \prod_{d=1}^{D} \frac{\prod_{k=1}^{K}\Gamma(n_{d,k}+\alpha_{d,k})}{\Gamma(\Sigma_{k=1}^{K}n_{d,k}+\alpha_{d,k})} \prod_{k=1}^{K} \frac{\prod_{v=1}^{V}\Gamma(n_{k,v}+\eta_{k,v})}{\Gamma(\Sigma_{v=1}^{V}n_{k,v}+\eta_{k,v})}$$

(13)

Se extrae aquellos términos que solo dependen de m, n (Geigle, 2017) (Ponweiser, 2012).

$$\propto \frac{\prod_{k=1}^{K}\Gamma(n_{d,k}+\alpha_{d,k})}{\Gamma(\Sigma_{k=1}^{K}n_{d,k}+\alpha_{d,k})} \times \prod_{k=1}^{K} \frac{\Gamma(n_{k,v}+\eta_{k,v})}{\Gamma(\Sigma_{v=1}^{V}n_{k,v}+\eta_{k,v})} \tag{14}$$

Ahora, se deja la siguiente propiedad (Geigle, 2017):

$$n_{k,d}^{-m,n} \begin{cases} n_{k,d}-1 & \text{si } d=m \text{ Y } k=z_{m,n} \\ n_{k,d} & \text{en caso contrario} \end{cases}$$

$$n_{k,v}^{-m,n} \begin{cases} n_{k,v}-1 & \text{si } k=z_{m,n} \text{ Y } v=z_{m,n} \\ n_{k,d} & \text{en caso contrario} \end{cases}$$

Se puede re escribir la ecuación numero 12 de la siguiente manera:

$$\propto \frac{(\prod_{i=1;i\neq k}\Gamma(n_{m,i}^{-(m,n)}+\alpha_i))\Gamma(1+n_{m,k}^{-(m,n)}+\alpha_k)}{\Gamma(1+\Sigma_{i=1}^{K}n_{m,i}^{-(m,n)}+\alpha_i)}\times$$
$$\prod_{i=1,i\neq k}\frac{\Gamma(n_{i,w_{m,n}^{-(m,n)}}+\eta_{w_{m,n}})}{\Gamma(\Sigma_{v=1}^{V}n_{k,v}^{-(m,n)}+\eta_{w_{m,n}})}\times$$
$$\frac{\Gamma(1+n_{k,w_{m,n}}^{-(m,n)}+\eta_{w_{m,n}})}{\Gamma(1+\Sigma_{v=1}^{V}n_{k,v}^{-(m,n)}+\eta_v)} \tag{15}$$

Ahora se usa la propiedad de la función gamma $\Gamma(1+x)=x\Gamma(x)$ para re escribir la ecuación 13 (Geigle, 2017):

$$\propto \frac{(\prod_{i=1;i\neq k}\Gamma(n_{m,i}^{-(m,n)}+\alpha_i))\Gamma(n_{m,k}^{-(m,n)}+\alpha_k)(n_{m,k}^{-(m,n)}+\alpha_k)}{\Gamma(\sum_{i=1}n_{m,i}^{-(m,n)}+\alpha_i)(\sum_{i=1}n_{m,i}^{-(m,n)}+\alpha_i)}\times$$
$$\prod_{i=1;i\neq k}\frac{\Gamma(n_{i,w}^{-(m,n)}+\eta_{w_{m,n}})}{\Gamma(\sum_{v=1}^{V}n_{i,v}^{-(m,n)}+\eta_v)}\times \tag{16}$$
$$\frac{\Gamma(n_{k,w_{m,n}}^{-(m,n)}+\eta_{w_{m,n}})(n_{k,w_{m,n}}^{-(m,n)}+\eta_{w_{m,n}})}{\Gamma(\sum_{v=1}^{V}n_{k,v}^{-(m,n)}+\eta_v)(\sum_{v=1}^{V}n_{k,v}^{-(m,n)}+\eta_v)}$$

Se procede a mover las funciones Γ dentro de los productos para que vuelvan a aparecer en todos los tópicos (Geigle, 2017).

$$\frac{\prod_{i=1}\Gamma(n_{m,i}^{-(-m,n)}+\alpha_i)}{\Gamma(\sum_{i=1}n_{m,i}^{*(m,n)}+\alpha_i)}\times\frac{n_{(m,k)}^{-(m,n)}+\alpha_k}{\sum_{i=1}n_{m,i}^{-(m,n)}+\alpha_i}\times\prod_{i=1}\frac{\Gamma(n_{i,w_{m,n}}^{-(m,n)}+\eta_{w_{m,n}})}{\Gamma(\sum_{v=1}n_{i,v}^{-(m,n)}+\eta_v)}\times\frac{n_{k,w_{m,n}}^{-(m,n)}+\eta_{w_{m,n}}}{\sum_{v=1}n_{k,v}^{-(m,n)}+\eta_v} \tag{17}$$

Finalmente, es posible desprenderse de los términos 1 y 3 de la ecuación 15, ya que no depende de $z_{m,n}=k$:

$$P(z_{m,n}=k|Z_{-(m,n)},w,\alpha,\eta)\propto\frac{n_{m,k}^{-(m,n)}+\alpha_k}{\sum_{i=1}n_{m,i}^{-(m,n)}+\alpha_i}\times\frac{n_{k,w_{m,n}}^{-(m,n)}+\eta_{w_{m,n}}}{\sum_{v=1}n_{k,v}^{-(m,n)}+\eta_v} \tag{18}$$

Para obtener las distribuciones de θ y β, simplemente podemos utilizar una estimación puntual basada en el estado actual de Z, en el algoritmo de MCMC (Geigle, 2017) (Speh , Muhic ,and Rupnik, 2021):

$$\theta_{m,k} = \frac{n_{m,k} + \alpha}{\sum_{j=1}^{K}(n_{m,j} + \alpha)} \tag{19}$$

$$\beta_{k,v} = \frac{n_{k,v} + \eta}{\sum_{i=1}^{V}(n_{k,i} + \eta)} \tag{20}$$

7.5. La Elección de los Hiper Parámetros.

La eleccion de los hiper parametro α y η puede ser una elección bastante compleja, ya que impacta el resultado del modelo por completo.

Se realizo un experimento (Griffiths, 2004), en el cual se eligieron los parametros $\eta = 0.01$ y $\alpha = 50/tópicos$). Se tenia un cantidad de mas de 28.000 documentos y mas de 20.000 palabras únicas y se intentaron la siguiente cantidad de tópicos: [50,100,200,300,400,500,600,1000]

Para la elección de α y η se busca principalmente resolver las siguientes preguntas (kedarps, 2019):

- Dado mi vocabulario de palabras, ¿espero que mis tópicos resultantes sean escasos? Para la mayoría de los casos, esto es cierto. Por lo tanto, normalmente el tópico anterior se elige para que sea escaso con $\eta < 1$.
- Dados los temas, ¿espero que la distribución de tópicos en cada documento sea escasa? Es decir, cada documento solo representa algunos tópicos. Si es así, entonces $\alpha < 1$.

El documento de (Griffiths T. , 2004) dice lo siguiente sobre la elección de los hiper parámetros: *Por lo tanto, el valor de η afecta la granularidad del modelo: un corpus de documentos se puede factorizar con sensatez en un conjunto de temas en varias escalas diferentes, y la escala particular evaluada por el modelo será establecida por η. Con documentos científicos, un valor grande de η llevaría al modelo a encontrar un número relativamente pequeño de temas, quizás a nivel de disciplinas científicas, mientras que valores más pequeños de η producirán más temas que aborden áreas específicas de investigación.*

En la librería de Python, Gensim, ambos parametros α y η son simétricamente predeterminados por: 1 / num-topicos a priori, y esta es la elección que se usara al momento de correr el algoritmo en el actual trabajo.

7.6. Ejemplo práctico en R

```
> library(dplyr)
> library(stringi)
> library(stringr)
> library(tm)
>
> words = stopwords(kind = "sp")
> words <- append(words,c("sino", "mas"))
>
> # se utilizaran 10 frases celebres en español
```

```
> #que se pueden considerar de las mas famosas.
> documentos <- c("Para trabajar basta estar convencido de una cosa: que trabajar es menos aburrido que divertirse"
+                 "Las guerras seguirán mientras el color de la piel siga siendo más importante que el de los ojos"
+                 "El mundo no está en peligro por las malas personas sino por aquellas que permiten la maldad",
+                 " No hay nada que un hombre no sea capaz de hacer cuando una mujer le mira",
+                 "Dar el ejemplo no es la principal manera de influir sobre los demás; es la única manera",
+                 "La mayor declaración de amor es la que no se hace; el hombre que siente mucho, habla poco",
+                 "Si das pescado a un hombre hambriento lo nutres durante una jornada. Si le enseñas a pescar, le
+                 "Estar preparado es importante, saber esperarlo es aún más, pero aprovechar el momento adecuado e
+                 "Lo que importa verdaderamente en la vida no son los objetivos que nos marcamos, sino los caminos
+                 "De todos los animales de la creación el hombre es el único que bebe sin tener sed, come sin tene
>
>
> # la siguiente, es una funcion que permite retirar
> # signos de puntuacion, acentos y pasa todas las letras
> # a minusculas
> # NOTA: en esta parte no se retiran stopwords,
> # ya que los datos no son demasiados.
> fun <- function(x){
+
+   df <- x %>%
+     tolower() %>%
+     str_replace_all("[^[:alnum:][:space:]]", "") %>%
+     stri_trans_general(id = "Latin-ASCII") %>%
+     str_replace_all("[^A-Za-z0-9 ]", "") %>%
+     removeWords(words = words) %>%
+     str_squish()
+
+   return(df)
+ }
>
>
> # se crea una lista nueva de frases o documentos
> # pero ya depurados.
> docs <- fun(documentos)
>
> # se separan las palabras de cada documento,
> # lo que se considera tokenizacion
> split_docs <- strsplit(docs, split = " ")
>
> split_docs_id <- split_docs
>
> # este es el vocabulario con las palabras unicas
> #que aparecen en todos los documentos.
> vocabulario <- unique(unlist(split_docs))
>
> # se reemplazan las palabras por ID's numericos.
> for( i in 1:length(split_docs) ) {
+   split_docs_id[[i]] <- match(split_docs_id[[i]], vocabulario)
+ }
```

```
> split_docs_id
[[1]]
[1] 1 2 3 4 1 5 6 7

[[2]]
[1]  8  9 10 11 12 13 14 15

[[3]]
[1] 16 17 18 19 20 21 22

[[4]]
[1] 23 24 25 26 27

[[5]]
[1] 28 29 30 31 32 33 34 31

[[6]]
[1] 35 36 37 38 23 39 40

[[7]]
 [1] 41 42 43 23 44 45 46 41 47 48 49 50 51

[[8]]
 [1] 52 14 53 54 55 56 57 58 59 51

[[9]]
[1] 60 61 51 62 63 64 65 66

[[10]]
 [1] 67 68 23 69 70 71 72 73 71 74 40 71 75

>
>
> # Un pequeño inconveniente de la asignación de
> # dirichlets latentes es que primero debe especificar
> # el número de clústeres. En otras palabras,
> # debe especificar el número de temas en los que
> # desea agrupar el conjunto de documentos por adelantado
> # (denotado por K). En nuestros casos usaremos 2.
> #
> # El primer paso del algoritmo es revisar cada
> # documento y asignar aleatoriamente cada palabra
> # del documento a uno de los temas K. Además de
> # generar esta lista de asignaciónde temas,
> # también crearemos una matriz palabra-tema,
> # que es el recuento de cada palabra que se
> # asigna a cada tema. Y una matriz documento-tema,
> # que es el número de palabras asignadas a cada
> # tema para cada documento (distribución de la
> # lista de asignación de temas). Usaremos las dos
```

```
> # matrices posteriores a lo largo del proceso
> # del algoritmo.
>
> # numero de clusters
> K <- 2
>
> # se iniciliza la matriz de conteos
> # pt : matriz palabra-topico
> pt <- matrix( 0, K, length(vocabulario) )
> colnames(pt) <- vocabulario
>
> # @at : lista de asignacion de topicos
> at <- lapply(split_docs_id, function(x) rep( 0, length(x) ) )
> names(at) <- paste0( "doc", 1:length(split_docs_id) )
>
> # @dt : conteos correspondientes al numero de
> # palabras a cada topico por cada documento
> dt <- matrix( 0, length(split_docs_id), K )
>
> set.seed(1)
> for( d in 1:length(split_docs_id)) {
+   # asignacion aleatoria de topico a la palabra w
+   for( w in 1:length(split_docs_id[[d]] ) ) {
+     at[[d]][w] <- sample(1:K, 1)
+
+     # se extrae el indice del topico, el id de
+     # la palabra, y se actualiza en la celda correspondiente
+     # de la matriz palabra-topico
+     ti <- at[[d]][w]
+     wi <- split_docs_id[[d]][w]
+     pt[ti, wi] <- pt[ti, wi] + 1
+   }
+
+   # se cuenta las palabras en el documento d,
+   # asignadas a cada topico k (t)
+   for( t in 1:K ) {
+     dt[d, t] <- sum( at[[d]] == t )
+   }
+ }
>
> # topicos asignados aleatoriamente a cada palabra
> print(at)
$doc1
[1] 1 2 1 1 2 1 1 1

$doc2
[1] 2 2 1 1 1 1 1 2

$doc3
[1] 2 2 2 1 1 1 1
```

```
$doc4
[1] 1 1 1 2 1

$doc5
[1] 1 2 2 2 1 2 1 1

$doc6
[1] 2 1 2 2 2 2 1

$doc7
 [1] 2 2 2 2 2 1 1 2 1 2 2 1 1

$doc8
 [1] 2 2 2 1 1 2 2 2 2 2

$doc9
[1] 2 1 2 2 2 2 1 1

$doc10
 [1] 1 2 2 1 1 2 2 2 1 1 1 2 1

> print(pt)
     trabajar basta convencido cosa menos aburrido divertirse
[1,]        1     0          1    1     1        1          1
[2,]        1     1          0    0     0        0          0
     guerras seguiran mientras color piel siga importante ojos
[1,]       0        0        1     1    1    1          1    0
[2,]       1        1        0     0    0    0          1    1
     mundo peligro malas personas aquellas permiten maldad
[1,]     0       0     0        1        1       1      1
[2,]     1       1     1        0        0       0      0
     hombre capaz hacer mujer mira dar ejemplo principal manera
[1,]      1     1     1     0    1   1       0         0      1
[2,]      3     0     0     1    0   0       1         1      1
     influir demas unica mayor declaracion amor hace siente
[1,]       1     0     1     0           1    0    0      0
[2,]       0     1     0     1           0    1    1      1
     habla si das pescado hambriento nutres jornada ensenas
[1,]     2  0   0       0          0      1       1       1
[2,]     0  2   1       1          1      0       0       0
     pescar nutriras toda vida preparado saber esperarlo aun
[1,]      0        0    1    1         0     0         1   1
[2,]      1        1    0    2         1     1         0   0
     aprovechar momento adecuado clave importa verdaderamente
[1,]          0       0        0     0       0              1
[2,]          1       1        1     1       1              0
     objetivos marcamos caminos seguimos lograrlo animales
[1,]         0        0       0        1        1        1
[2,]         1        1       1        0        0        0
```

```
     creacion unico bebe tener sed come hambre decir
[1,]        0     1    1     1   0    0      1     1
[2,]        1     0    0     2   1    1      0     0
> print(dt)
      [,1] [,2]
 [1,]    6    2
 [2,]    5    3
 [3,]    4    3
 [4,]    4    1
 [5,]    4    4
 [6,]    2    5
 [7,]    5    8
 [8,]    2    8
 [9,]    3    5
[10,]    7    6
>
> # La anterior asignacion ya devueleve las
> # representaciones de topicos de todos los documentos,
> # asi como las distribuciones de palabras (similar
> # al algoritmo LSA), pero la calidad deja mucho
> # que desear. Por esto, se utiliza el
> # muestreador de Gibbs.
>
> # parametros
> alpha <- 1
> eta <- 1
>
> # los primeros topicos asignados a la primera
> # palabra del primer documento
> # y su correspondiente ID
> t0  <- at[[1]][1]
> wid <- split_docs_id[[1]][1]
>
> # z_-(m,n) significa que no se incluye la parte
> # (m,n) de w, en la matriz de conteo palabra-topico
> # y documento-topicom cuando se esta muestreando
> # para la parte w solo se deja la asignacion
> # del topico de todas las otras componentes para el
> # primer documento
> dt[1, t0] <- dt[1, t0] - 1
> pt[t0, wid] <- pt[t0, wid] - 1
>
>
> # se calcula el lado izquierdo y el derecho de
> # la ecuacion proporcional de Gibbs
> izquierda  <- ( pt[, wid] + eta ) / ( rowSums(pt) + length(vocabulario) * eta )
> derecha <- ( dt[1, ] + alpha ) / ( sum( dt[1, ] ) + K * alpha )
>
> # se escoge un nuevo topico para la primera
> # palabra en el primer documento el argumento
```

```
> # prob se provee para pasar un vector de pesos
> # a la funcion sample, y obtener los elementos
> # del vector que se esta muestreando.
> # No necesariamente deben sumar uno, pero no
> # deben ser negativos y no muchos deben ser cero.
>
> t1 <- sample(1:K, 1, prob = izquierda*derecha)
> t1
[1] 1
>
> pt[1,]
      trabajar          basta     convencido          cosa
             0              0              1             1
         menos       aburrido     divertirse        guerras
             1              1              1             0
      seguiran       mientras          color          piel
             0              1              1             1
          siga     importante           ojos         mundo
             1              1              0             0
       peligro          malas       personas       aquellas
             0              0              1             1
      permiten        maldad         hombre          capaz
             1              1              1             1
         hacer          mujer          mira            dar
             1              0              1             1
       ejemplo      principal         manera       influir
             0              0              1             1
         demas          unica         mayor    declaracion
             0              1              0             1
          amor           hace        siente          habla
             0              0              0             2
            si            das       pescado     hambriento
             0              0              0             0
        nutres        jornada        ensenas         pescar
             1              1              1             0
      nutriras           toda          vida      preparado
             0              1              1             0
         saber      esperarlo           aun     aprovechar
             0              1              1             0
       momento       adecuado         clave        importa
             0              0              0             0
verdaderamente      objetivos       marcamos        caminos
             1              0              0             0
      seguimos       lograrlo       animales       creacion
             1              1              1             0
         unico          bebe          tener           sed
             1              1              1             0
          come         hambre          decir
             0              1              1
>
```

```
>
> modelo_lda <- function( docs, vocab, K, alpha, eta, iteraciones) {
+   # se iniciliza la matriz de conteos
+   # wt : matriz palabra-topico
+   wt <- matrix( 0, K, length(vocab) )
+   colnames(wt) <- vocab
+
+   # @ta : lista de asignacion de topicos
+   ta <- lapply( docs, function(x) rep( 0, length(x) ) )
+   names(ta) <- paste0( "doc", 1:length(docs) )
+
+   # @dt : conteos correspondientes al numero
+   #de palabras a cada topico por cada documento
+   dt <- matrix( 0, length(docs), K )
+
+   for( d in 1:length(docs) )
+   {
+     # aleatoriamente asignar topicos a la palabra w
+     for( w in 1:length( docs[[d]] ) )
+     {
+       ta[[d]][w] <- sample( 1:K, 1 )
+
+       # extraer la posicion del topico,
+       #el id de la palabra y actualizar la fila-columna
+       #correspondiente en la posisicion de
+       #la matriz wt correspondiente
+       ti <- ta[[d]][w]
+       wi <- docs[[d]][w]
+       wt[ ti, wi ] <- wt[ ti, wi ] + 1
+     }
+
+     # contar las palabras en el documento d,
+     # asignado a cada topico t
+     for( t in 1:K )
+       dt[ d, t ] <- sum( ta[[d]] == t )
+   }
+
+   # para cada paso i a traves del corpus,
+   # el numero de iteraciones especificado,
+   for( i in 1:iteraciones )
+   {
+     # para cada documento
+     for( d in 1:length(docs) )
+     {
+       # para cada palabra
+       for( w in 1:length( docs[[d]] ) )
+       {
+         t0  <- ta[[d]][w]
+         wid <- docs[[d]][w]
+
```

```
+         dt[ d, t0 ]   <- dt[ d, t0 ] - 1
+         wt[ t0, wid ] <- wt[ t0, wid ] - 1
+
+         left  <- ( wt[ , wid ] + eta ) / ( rowSums(wt) + length(vocab) * eta )
+         right <- ( dt[ d, ] + alpha ) / ( sum( dt[ d, ] ) + K * alpha )
+
+         t1 <- sample( 1:K, 1, prob = left * right )
+
+         # acualizar la lista de asigancion
+         # de topicos (ta) con los nuevos topicos
+         # muestreados para cada indice w.
+         # se actualiza tambien las matrices
+         #wt (palabra-topico) y dt (docuemnto-topico)
+         # con los nuevos topicos remuestreados
+         #para cada indice w
+         ta[[d]][w] <- t1
+         dt[ d, t1 ]   <- dt[ d, t1 ] + 1
+         wt[ t1, wid ] <- wt[ t1, wid ] + 1
+
+         # codigo opcional: se examina cuando
+         #los topicos se intercambian
+         # if( t0 != t1 )
+         #  print( paste0( "doc:", d, " token:" ,
+         #  w, " topic:", t0, "=>", t1 ) )
+       }
+     }
+   }
+
+   return( list( wt = wt, dt = dt ) )
+ }
>
>
> K <- 3 # topicos
> alpha <- 1/K # words
> eta <- 1/K # topicos
> iteraciones <- 1000
>
> set.seed(1)
> lda1 <- modelo_lda( docs = split_docs_id, vocab = vocabulario,
+                     K = K, alpha = alpha, eta = eta, iteraciones = iteraciones )
> lda1
$wt
     trabajar basta convencido cosa menos aburrido divertirse
[1,]        0     0          0    0     1        0          0
[2,]        2     1          0    1     0        1          1
[3,]        0     0          1    0     0        0          0
     guerras seguiran mientras color piel siga importante ojos
[1,]       0        0        0     0    0    0          0    0
[2,]       1        1        1     1    1    1          1    1
[3,]       0        0        0     0    0    0          1    0
```

```
     mundo peligro malas personas aquellas permiten maldad
[1,]     0       0     0        0        0        0      0
[2,]     0       0     0        0        0        0      0
[3,]     1       1     1        1        1        1      1
     hombre capaz hacer mujer mira dar ejemplo principal manera
[1,]      3     0     0     1    0   0       0         0      0
[2,]      0     1     1     0    0   1       1         1      2
[3,]      1     0     0     0    1   0       0         0      0
     influir demas unica mayor declaracion amor hace siente
[1,]       0     0     0     1           1    1    0      0
[2,]       0     1     1     0           0    0    1      1
[3,]       1     0     0     0           0    0    0      0
     habla si das pescado hambriento nutres jornada ensenas
[1,]     2  2   1       1          1      1       1       1
[2,]     0  0   0       0          0      0       0       0
[3,]     0  0   0       0          0      0       0       0
     pescar nutriras toda vida preparado saber esperarlo aun
[1,]      1        1    1    0         0     0         0   0
[2,]      0        0    0    1         0     0         0   0
[3,]      0        0    0    2         1     1         1   1
     aprovechar momento adecuado clave importa verdaderamente
[1,]          0       0        0     0       1               1
[2,]          0       0        0     0       0               0
[3,]          1       1        1     1       0               0
     objetivos marcamos caminos seguimos lograrlo animales
[1,]         0        0       0        0        1        1
[2,]         1        0       1        0        0        0
[3,]         0        1       0        1        0        0
     creacion unico bebe tener sed come hambre decir
[1,]        1     0    1     3   1    1      1     1
[2,]        0     1    0     0   0    0      0     0
[3,]        0     0    0     0   0    0      0     0

$dt
      [,1] [,2] [,3]
 [1,]    1    6    1
 [2,]    0    8    0
 [3,]    0    0    7
 [4,]    1    2    2
 [5,]    0    7    1
 [6,]    5    2    0
 [7,]   12    0    1
 [8,]    0    0   10
 [9,]    3    3    2
[10,]   12    1    0

>
> # Una vez que hayamos terminado de aprender
> # los temas para 1000 iteraciones, podemos
> # usar las matrices de recuento para obtener
```

```
> # la distribución palabra-temay la
> # distribución documento-tema.
> # Para calcular la probabilidad de la palabra
> # dada del tema:
>
> # probabilidad del tema para cada palabra
> ( beta <- ( lda1$wt + eta ) / ( rowSums(lda1$wt) + length(vocabulario) * eta ) )
        trabajar       basta  convencido        cosa       menos
[1,] 0.005649718 0.005649718 0.005649718 0.005649718 0.022598870
[2,] 0.043209877 0.024691358 0.006172840 0.024691358 0.006172840
[3,] 0.006802721 0.006802721 0.027210884 0.006802721 0.006802721
        aburrido  divertirse     guerras    seguiran    mientras
[1,] 0.005649718 0.005649718 0.005649718 0.005649718 0.005649718
[2,] 0.024691358 0.024691358 0.024691358 0.024691358 0.024691358
[3,] 0.006802721 0.006802721 0.006802721 0.006802721 0.006802721
           color        piel        siga  importante        ojos
[1,] 0.005649718 0.005649718 0.005649718 0.005649718 0.005649718
[2,] 0.024691358 0.024691358 0.024691358 0.024691358 0.024691358
[3,] 0.006802721 0.006802721 0.006802721 0.027210884 0.006802721
           mundo     peligro       malas    personas    aquellas
[1,] 0.005649718 0.005649718 0.005649718 0.005649718 0.005649718
[2,] 0.006172840 0.006172840 0.006172840 0.006172840 0.006172840
[3,] 0.027210884 0.027210884 0.027210884 0.027210884 0.027210884
        permiten      maldad     hombre       capaz       hacer
[1,] 0.005649718 0.005649718 0.05649718 0.005649718 0.005649718
[2,] 0.006172840 0.006172840 0.00617284 0.024691358 0.024691358
[3,] 0.027210884 0.027210884 0.02721088 0.006802721 0.006802721
           mujer        mira         dar     ejemplo   principal
[1,] 0.022598870 0.005649718 0.005649718 0.005649718 0.005649718
[2,] 0.006172840 0.006172840 0.024691358 0.024691358 0.024691358
[3,] 0.006802721 0.027210884 0.006802721 0.006802721 0.006802721
         manera     influir       demas       unica       mayor
[1,] 0.005649718 0.005649718 0.005649718 0.005649718 0.022598870
[2,] 0.043209877 0.006172840 0.024691358 0.024691358 0.006172840
[3,] 0.006802721 0.027210884 0.006802721 0.006802721 0.006802721
     declaracion        amor        hace      siente       habla
[1,] 0.022598870 0.022598870 0.005649718 0.005649718 0.039548023
[2,] 0.006172840 0.006172840 0.024691358 0.024691358 0.006172840
[3,] 0.006802721 0.006802721 0.006802721 0.006802721 0.006802721
              si         das     pescado  hambriento      nutres
[1,] 0.039548023 0.022598870 0.022598870 0.022598870 0.022598870
[2,] 0.006172840 0.006172840 0.006172840 0.006172840 0.006172840
[3,] 0.006802721 0.006802721 0.006802721 0.006802721 0.006802721
         jornada     ensenas      pescar    nutriras        toda
[1,] 0.022598870 0.022598870 0.022598870 0.022598870 0.022598870
[2,] 0.006172840 0.006172840 0.006172840 0.006172840 0.006172840
[3,] 0.006802721 0.006802721 0.006802721 0.006802721 0.006802721
            vida   preparado       saber   esperarlo         aun
[1,] 0.005649718 0.005649718 0.005649718 0.005649718 0.005649718
[2,] 0.024691358 0.006172840 0.006172840 0.006172840 0.006172840
```

```
[3,] 0.047619048 0.027210884 0.027210884 0.027210884 0.027210884
      aprovechar     momento    adecuado       clave     importa
[1,] 0.005649718 0.005649718 0.005649718 0.005649718 0.022598870
[2,] 0.006172840 0.006172840 0.006172840 0.006172840 0.006172840
[3,] 0.027210884 0.027210884 0.027210884 0.027210884 0.006802721
     verdaderamente   objetivos    marcamos     caminos
[1,]    0.022598870 0.005649718 0.005649718 0.005649718
[2,]    0.006172840 0.024691358 0.006172840 0.024691358
[3,]    0.006802721 0.006802721 0.027210884 0.006802721
        seguimos    lograrlo    animales    creacion       unico
[1,] 0.005649718 0.022598870 0.022598870 0.022598870 0.005649718
[2,] 0.006172840 0.006172840 0.006172840 0.006172840 0.024691358
[3,] 0.027210884 0.006802721 0.006802721 0.006802721 0.006802721
            bebe       tener         sed        come      hambre
[1,] 0.022598870 0.056497175 0.022598870 0.022598870 0.022598870
[2,] 0.006172840 0.006172840 0.006172840 0.006172840 0.006172840
[3,] 0.006802721 0.006802721 0.006802721 0.006802721 0.006802721
           decir
[1,] 0.022598870
[2,] 0.006172840
[3,] 0.006802721
>
> # probabilidad del tema para cada documento
> ( theta <- ( lda1$dt + alpha ) / ( rowSums(lda1$dt) + K * alpha ) )
             [,1]       [,2]       [,3]
 [1,] 0.14814815 0.70370370 0.14814815
 [2,] 0.03703704 0.92592593 0.03703704
 [3,] 0.04166667 0.04166667 0.91666667
 [4,] 0.22222222 0.38888889 0.38888889
 [5,] 0.03703704 0.81481481 0.14814815
 [6,] 0.66666667 0.29166667 0.04166667
 [7,] 0.88095238 0.02380952 0.09523810
 [8,] 0.03030303 0.03030303 0.93939394
 [9,] 0.37037037 0.37037037 0.25925926
[10,] 0.88095238 0.09523810 0.02380952
>
>
>
> # Recordemos que LDA asume que cada documento
> # es una mezcla de todos los temas, por lo
> # que después de calcular la probabilidad
> # de que cada documento pertenezca a cada
> # tema (lo mismo ocurre con word y topic)
> # podemos usar esta información para ver
> # a qué tema pertenece cada documento y
> # las palabras más posibles que están
> # asociadas con cada tema.
>
>
>
```

```
> # asignacion del topico para cada documento,
> # el que tenga mayor probabilidad
> topico <- apply(theta, 1, which.max)
>
> # posibles palabras de cada topico
> #
> terminos <- function(beta, n) {
+   termino <- matrix(0, n, K)
+   for( p in 1:nrow(beta) ) {
+     termino[, p] <- names( sort( beta[p, ], decreasing = TRUE )[1:n] )
+   }
+   return(termino)
+ }
> termino <- terminos(beta = beta, n = 3)
>
> list( original_text = documentos[topico == 1], words = termino[, 1] )
$original_text
[1] "La mayor declaración de amor es la que no se hace; el hombre que siente mucho, habla poco"
[2] "Si das pescado a un hombre hambriento lo nutres durante una jornada. Si le enseñas a pescar, le nutrirás toda
[3] "Lo que importa verdaderamente en la vida no son los objetivos que nos marcamos, sino los caminos que seguimos
[4] "De todos los animales de la creación el hombre es el único que bebe sin tener sed, come sin tener hambre y hab

$words
[1] "hombre" "tener"  "habla"

>
> list( original_text = documentos[topico == 2], words = termino[, 2] )
$original_text
[1] "Para trabajar basta estar convencido de una cosa: que trabajar es menos aburrido que divertirse"
[2] "Las guerras seguirán mientras el color de la piel siga siendo más importante que el de los ojos"
[3] " No hay nada que un hombre no sea capaz de hacer cuando una mujer le mira"
[4] "Dar el ejemplo no es la principal manera de influir sobre los demás; es la única manera"

$words
[1] "trabajar" "manera"   "basta"
```

8. Metodología

En el presente proyecto se enfoca en dos algoritmos principalmente, los cuales son el algoritmo de Bayes Ingenuo, y el algoritmo de Asignación de Dirichlet Latente. ¿Cuáles son las principales diferencias de estos dos algoritmos? Bueno, aunque los dos algoritmos tienen en común que son basados en los principios de la estadística bayesiana, los dos son tipos de aprendizaje diferentes: el algoritmo Ingenuo de Bayes es aprendizaje supervisado, por lo cual se necesita dividir los datos en test y prueba, mientras que el algoritmo de análisis Latente Semántico es aprendizaje no supervisado, con lo cual el mismo algoritmo se puede evaluar de una manera distinta a la acostumbrada con las matrices de confusión.

8.1. Obtención de los Datos

Para el presente proyecto, los datos que se usaran provienen de dos fuentes: datos de Twitter en tiempo real, gracias a la API de Twitter Developers, y tweets que fueron recolectados de distintos países de habla hispana, para los talleres de Análisis Semántico (TASS), que son workshops organizados por la Sociedad Española de Procesamiento del Lenguaje Natural.

Figura 8: Fuentes de datos
Fuente: elaboración propia

- **Twitter API**: La API de Twitter, es una herramienta desarrollada para los desarrolladores de productos, empresas, y usuarios en general. Las API son herramientas especiales con las cuales los softwares informáticos se comunican entre ellos, para intercambiar información, pero depende del nivel de seguridad establecido por cada software para realizar estos intercambios. Cuando una persona quiere hacer uso de la API de Twitter, debe registrar una Aplicación. Para registrar una aplicación debe primero tener una cuenta activa de usuario de Twitter. De forma predeterminada, se tiene acceso únicamente a la información pública de Twitter. Datos como el envío o recepción de mensajes directos requiere permisos especiales. Los datos que se recopilan a través de la API de Twitter, y del stremming de los datos, son variables como el texto del tweet, la fecha, el usuario, la locación (si el usuario tiene habilitada esta opción), y el hashtag. Además de esto, se utiliza las funciones de filtro, para acceder a los datos que se generan en las ciudades de: Bogotá, Medellín, Cali, Cúcuta, Pereira y Cartagena.
- **TASS**: Los workshop o talleres de "Análisis Semántico de la SEPLN" (TASS), fueron las primeras tareas compartidas sobre análisis de sentimientos, utilizando datos de Twitter, aunque su uso y aplicabilidad se pueden extender a muchas más plataformas metas, como el análisis de encuestas abiertas, PQR's, etc. Este evento tuvo su primera edición en el año 2012, y hasta la fecha se han realizado todos los

años, con una breve intermitencia en el año 2020 debido a la pandemia de Covid 19. Los datos además de los tweets, vienen con otras variables como lo son UserID, que corresponde al ID del usuario de Twitter, Fecha, y Polaridad. Polaridad es la variable más importante, que tiene tres categorías las cuales son N, P y NEU (Negativos, Positivos y Neutros), e indican el sentimiento del tweet. NOTA: Para el presente trabajo, solo es de interés las polaridades P y N, por lo cual se realiza un filtro para dejar solo estas características.

8.2. Procesamiento de los datos

Para el pre preprocesamiento de los datos, se tienen en cuenta los siguientes procesos:

1. Primero se realiza un proceso de limpieza, en la cual se remueven de los tweets cosas como números, caracteres especiales, emoticones puntos, comas y demás, y luego pasar todas las palabras a forma minúscula.
2. Posteriormente se realiza el proceso de tokenización, el cual consiste en separar palabra por palabra cada tweet, en una lista para cada tweet.
3. Después de esto, se remueven las stopwords (palabras no relevantes como "el", "las", "con"); además de estas palabras StopWords en español, se pueden agregar más palabras que no se deseen observar.
4. En este punto, se realiza dos diferentes procesos para cada conjunto de datos. Para los datos de Twitter obtenidos a través de TASS, se procede a realizar el proceso de Stemming, que fue descrito con anterioridad, mientras que, para los tweets obtenidos a través de la API de Twitter, se realiza el proceso de Lematización. La razón de realizar dos procesos distintos, procede a que el algoritmo de Bayes Ingenuo tiene un tiempo de ejecución más prolongado dado por su naturaleza de algoritmo de aprendizaje supervisado, y el proceso de lematización requiere más potencia de máquina que el proceso de Stemming, por lo cual se decidió dejar el proceso de lematización para el algoritmo de Análisis Latente semántico, y el de Stemming para el algoritmo de Bayes Ingenuo. Otra razón de esto, es que, en el algoritmo de Bayes Ingenuo, se pretende observar la polaridad de los tweets, no las palabras de los tweets en si, en el algoritmo de Análisis Latente semántico si se procede a observar las palabras, por lo cual es mucho más comprensible el proceso de lematización por encima del Stemming.

8.3. Modelado

En esta etapa, se realiza el proceso de ajuste de los modelos, por lo cual se siguen dos procesos diferentes para el ajuste de cada uno de los modelos.

8.3.1. Bayes Ingenuo

Para el modelo de Bayes Ingenuo, se tiene una base de aproximadamente 3500 tweets recopilados de diferentes países de habla hispana, los cuales después de realizarles un pre procesamiento para arreglar la base de datos, y dejarla de la forma predeterminada, se realiza el proceso de partición de la base de datos en los que es la parte de entrenamiento

y la parte de prueba. Posteriormente, se ajusta el modelo de Bayes Ingenuo, utilizando el módulo multinomial, que es el más apropiado para el modelado de variables multinomiales discretas, como lo es el conteo de palabras, y que en la práctica, funciona muy bien para conteos parciales como los de TF-IDF (Term Frequency – Inverse Document Frequency).

8.3.2. Asignación de Dirichlet Latente.

Para este algoritmo, se tiene más de 600 mil tweets obtenidos en las principales capitales de Colombia, gracias a las opciones de filtros que la API de Twitter permite aplicar. Después de el pre procesamiento de los datos correspondiente, se procede a crear un diccionario de los tweets procesados, lo que genera un diccionario de palabras, similar a un mapeo de palabras con sus correspondientes ID's. Después de esto, se procede a obtener la bolsa de palabras del diccionario generado con anterioridad. Esta bolsa de palabras genera un resultado en forma de tupla, el cual contiene *token_id*, identificación de la palabra, y *token_count* que corresponde a una asignación de contabilización de la palabra. NOTA: Se asume en este proceso que cada palabra es un carácter token izado y normalizado (Unicode o UTF-8). Posteriormente, se entrena el modelo con el modelo de Gensim, ldamulticore que tiene usa todos los nucleos del computador para realizar el ajuste del modelo, lo cual brinda una clara ganancia en la velocidad del algoritmo. Posteriormente, se pasa los parámetros al modelo como el corpus, que hace referencia a una matriz dispersa, y que en este caso corresponde a la bolsa de palabras creada. Luego la función pide el diccionario de palabras que se creó también con anterioridad, así como el número de tópicos que se desea generar en este modelo. También se agrega el parámetro de *random_state* , lo cual permite pasar una semilla al modelo y generar resultados reproducibles.

8.4. Métricas de Evaluación

En cuanto a las métricas de evaluación, se tienen en cuenta los siguientes tipos de métricas:

- Matriz de Confusión: *'Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real., o sea en términos prácticos nos permite ver qué tipos de aciertos y errores está teniendo nuestro modelo a la hora de pasar por el proceso de aprendizaje con los datos.'* Tomado de: (Barrios Arce, 2019).
- Curva ROC: Se recomienda el uso de la curva ROC en problemas de clasificación binaria. Esta métrica muestra el número de ejemplos positivos clasificados correctamente contra el número de ejemplos negativos clasificado incorrectamente (Davis, 2006).
- Curva Precision-Recall: esta métrica es usualmente usada en problemas en los cuales se tiene un gran sesgo de la información, así como un des balanceo de los datos (Davis, 2006).
- Métrica C_v Coherence Measure: perteneciente a las métricas de Coherencia de tópicos, se han discutido ampliamente en el estudio de procesamiento del lenguaje natural para cuantificar la implementación y el ajuste de piezas de información textual unidas entre sí (Roder, Both, Rosner, Et.Al, 2014).

9. Implementación

9.1. Librerías y lenguajes de Programación

Para el presente trabajo, se realizo una implementación de diferentes librerías para el lenguaje de programación Python. Unas de las principales librerías usadas fueron:

- NLTK: versión = 3.5 Esta es libraría es un conjunto de bibliotecas especializadas para el procesamiento del lenguaje natural, con un enfoque tanto simbólico como estadístico.
- Gensim: versión = 3.8.0 Es una librería de código abierto especializada en modelos no supervisados para temas de procesamiento del lenguaje natural, y modelado de tópicos, escrito en lenguaje Python.
- Spacy: versión = 2.3.5 Es una librería para el procesamiento de textos, desarrollado para Python en la universidad MIT, y lanzado en el año 2015.
- Scikit-learn: versión = 0.23.5 Es una de las principales librerías para el desarrollo de modelos de aprendizaje supervisado y no supervisado, además de poseer una gran ayuda para el usuario en cuanto a temas estadísticos, matemáticos, de programación y demás ciencias relacionadas. Fue escrito en lenguaje Python, C y C++.
- Pandas: versión = 1.2.4 Es una biblioteca de código abierto, ideada como una extensión a la librería Numpy, especializada en la manipulación, creación y análisis de datos para el lenguaje Python.
- Numpy: versión = 1.20.1 Es una librería que provee herramientas para la creación y manipulación de vectores y matrices numéricas, incluso de gran tamaño. También posee un gran abanico de funciones matemáticas y estadísticas para operar matrices, vectores y escalares. Esta escrita en los lenguajes Python, C y Fortran.
- Python: versión = 3.7 Es un lenguaje de programación interpretado, lo que significa, que a diferencia de otros lenguajes de programación, su código es bastante mas legible y entendible en comparación a otros lenguajes. Este es un lenguaje de programación que en los últimos años, ha tomado una gran acogida para los proyectos de Minería de Datos, análisis estadístico, series de tiempo, análisis matemático, y análisis financieros. Es un lenguaje de programación multi paradigma, puesto que tiene soporte tanto para programación imperativa, programación orientada a objetos y programación funcional.

9.2. Herramientas y Hardware

El desarrollo de este trabajo se realizo utilizando una computadora de mesa, con las siguientes especificaciones técnicas:

- Windows 10 Home Basic de 64 bits.
- Procesador Intel Core i7 de decima generación 10510U, caché de 8 MB, velocidad de hasta 4.9 GHz.
- Disco duro de Estado Solido de 520 GB.
- Memoria DDR4 3600hz 16GB

10. Aplicación

Se tienen listos los datos de Twitter provenientes de TASS, que son descargados en formato XML. Después se suben al entorno de Python y se concatenan. Después de concatenar datos de Twitter, se obtuvieron los siguientes datos preliminares sobre las palabras o unigramas mas frecuentes. Se puede observar las 30 palabras mas frecuentes, después de retirar todas las stopwords:

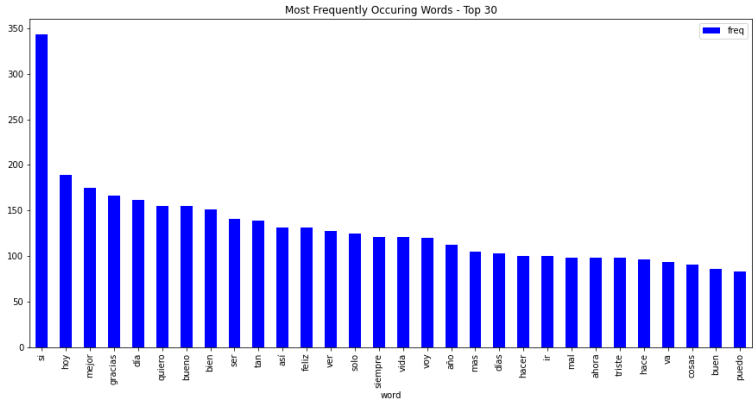


Figura 9: Histograma palabras más frecuentes
Fuente: elaboración propia

Además de observar las palabras más comunes de los datos de tweets, de los talleres de TASS, es interesante observar las polaridades de las palabras. Esto quiere decir, observar un conteo de los tweets por positivos o negativos.

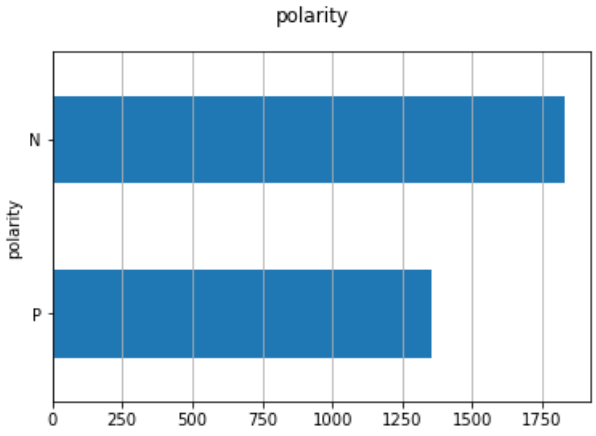


Figura 10: Histograma por polaridad
Fuente: elaboración propia

Se puede observar que se tienen aproximadamente 1300 tweets con una polaridad positiva, y 1800 tweets con una polaridad negativa. Ya que los datos están un poco desbalanceados, se utilizará también la métrica de curva precision-recall y la curva ROC, que ayudan a observar de una mejor manera la calidad del modelo en datos no balanceados. Se puede observar un ejemplo de los tweets de TASS en la tabla 1

11. siiii fue super gracioso teníamos que habernos sacado una foto
toNi_end seria mejor que dejasen de emitir esa basura ya hay que evolucionar para bien y eso
jonoro96 te mandaria a comprarte un burro, pero no creo que hayan tiendas abiertas ahora
Mg y pongo un adjetivo super repelente a vuestro nombre
hywzz la voz de María al final me mata JAJAJSJAJAJAJ quilla y yo que EduGalan14 Jaja. Es de bien nacidos ser agradecidos que diría aquel.Tiene el récord de puntos del VCF. Creo q cumplió con lo que había.
misscatingcat Bueno, al menos has acabado la noche con una hermosa vista Ami tambien me gusta verlos por la noche
mendescodes heroinseb enserio cuando habéis dicho que los froot loops están malos me ha dolido...
Sakura_Abril Ow Bueno, no pasa nada, cuando puedas confirmarlo, estoy aquí Y si no pudieras de cosplay pero sí a la expo, +
Aunque pensaba que no podría hacerlo hasta el mes de octubre, el lunes próximo volveré a ver a mi familia de # Mataró #felicidad

Tabla 1: Ejemplo tweets TASS

Es de interés también observar datos de los que se han extraído en tiempo real gracias a la API de Twitter. En la tabla 2 se puede observar un ejemplo de los tweets, al azar, que han sido recopilados en tiempo real desde junio del 2021 hasta octubre de 2021.

Clamor Nacional.
La selección colombiana ce rato no me ilusiona, estoy de acuerdo con jorgebermudezh #ESPNFShowColombia EneidaDazRomer1 sinamosindios WRadioColombia ClaudiaLopez Asumo q si, por eso no ando presumiendo el estudio q tengo, más si lo hago con mi inteligencia.
EVernacci ¿Le he dicho alguna vez que es usted mi héroe? ¡Menudo aguante!
Ellos no aceptaron? La que no asistió fue usted.
¡Nuevo juguete!
jflafaurie Y le falta una Dr. Billetes de 1 millón de pesos. Estamos jodidos. Por favor anime a la Dra. Cabal, ojala se lanzará pero urgente.
El #Quindío que hoy celebra sus 55 años como departamento, ha demostrado ser tierra acogedora, dueña de hermosos paisajes y cuna de cafetales que conforman inolvidables paisajes; atractivos que las #AgenciasdeViajes recomiendan, sin falta.
MarceLaALarcon Yo
Grumete40 lafm Total, las cosas que dice, y se siente con aura de sabiduría, sin palabras!!!
https://t.co/Rx17oQoOtP
Acaba de publicar un video en Colombia https://t.co/QeWA5TtzEq
Muy simple!
Vamos a ayudarla Tweet1A
Nos matamos y lo vemos normal por lograr lo gue gueremos
voleusedefleurs Primero, te mando buena energía para que te alivies pronto.
Matador000 CarlosCarrilloA Jajajaja es la única manera de hacer un metro elevado en Colombia por eso anunciaron; el Cerdo y el En-ríquito chiquito con tetes, la construcción de la segunda línea de Metro y va a ser subterráneo, osease que ya enterraron ese proyecto, lo mismo que el cable de San Cristóbal.
Nicaz_
FelipeDuque22 luismorales0621 La de él sí
JavAlexander2 _SebasabeS_ También me hubiera roto el alma :(
ClaudiaLopez Que desgraciada alcaldesa tenemos.
petrogustavo tan bipolar como ClaudiaLopez

Tabla 2: Ejemplo tweets API

Además de esto, se puede observar un gráfico de unigramas y bigramas, de los datos de Twitter recolectados con la API de la misma aplicación.

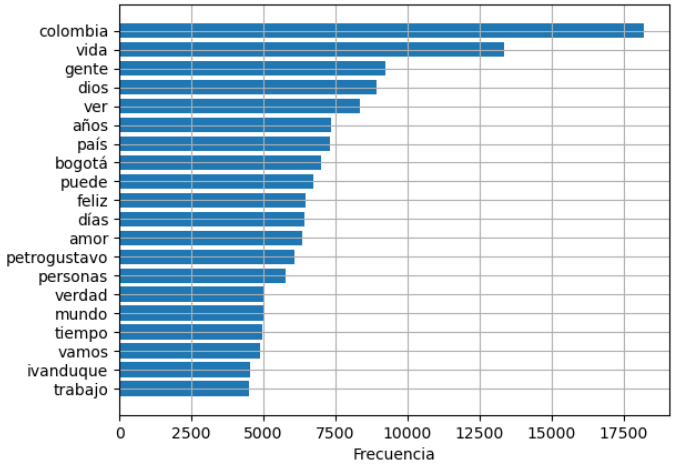


Figura 11: Unigramas más comunes
Fuente: elaboración propia

Se puede observar que "Colombia" es la palabra más común en los tweets. Algo muy interesante es que los personajes políticos que mas son nombrados en Twitter, en las ciudades capitales de Colombia, de donde se recolectaron estos datos, son "petrogustavo" y "ivanduque", haciendo alusión al senador Gustavo Petro, y al presidente Ivan Duque respectivamente. Estas palabras fueron tomadas después de eliminar algunas que no eran de total relevancia, y que se pueden encontrar en el script.

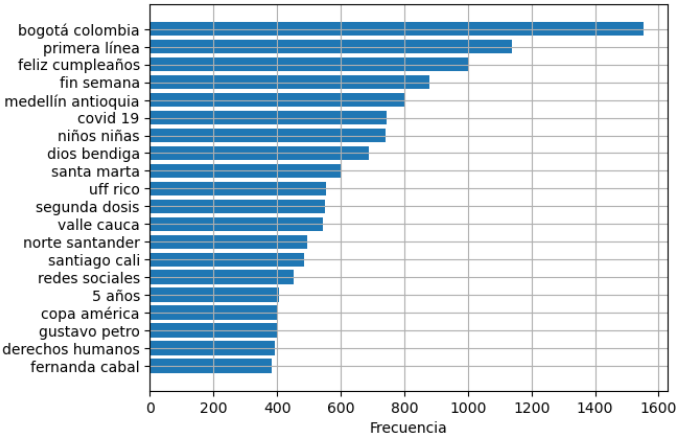


Figura 12: Bigramas más comunes
Fuente: elaboración propia

En cuanto a los bigramas mas comunes, se puede observar que "Bogotá colombia", es el más común del set de datos recolectados, haciendo alusión a la ciudad capital de Bogotá.

Es de recalcar que igual que la gráfica de unigramas de la ilustración, en este caso también se eliminaron algunas palabras sin relevancia previo a la realización del gráfico.

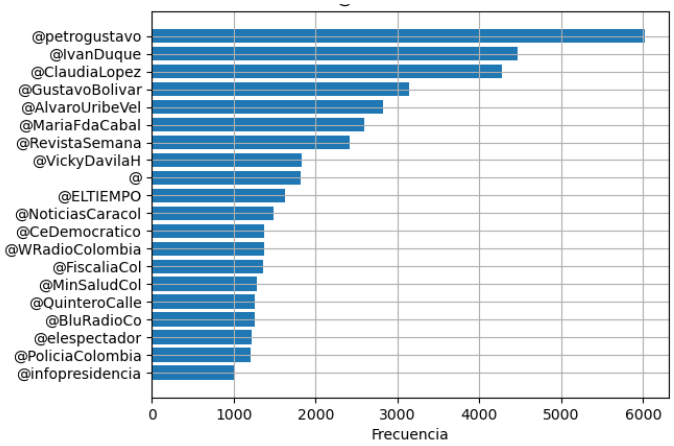


Figura 13: arrobas más comunes
Fuente: elaboración propia

Entre los arrobas más comunes, dentro de los datos recolectados de la API de Twitter, se destaca que son los de personalidades políticas los mas comunes, y no los de personalidades del mundo del cine, músicos, personalidades, influencers o demás personajes famosos. También es de destacar que "El Tiempo Noticias" parece ser el arroba más famoso entre los medios de comunicación.

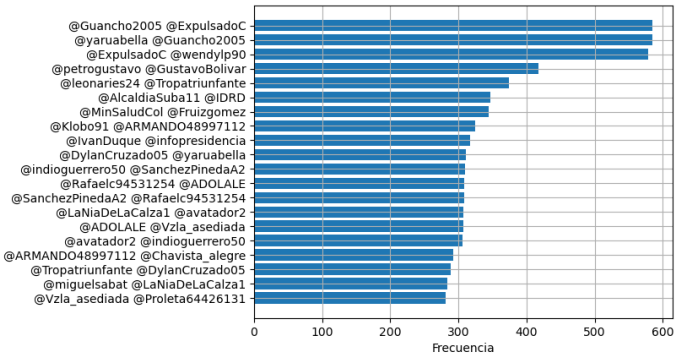


Figura 14: arrobas compuestos más comunes
Fuente: elaboración propia

Ahora, de los arrobas compuestos, se destaca el de @Guancho2005 y el @yaruabella, que son twitteros venezolanos bastante activos, pero que están haciendo sus publicaciones en la red social Twitter, desde territorio colombiano. Es de destacar que @petrogustavo parece estar casi siempre acompañado de @GustavoBolivar.

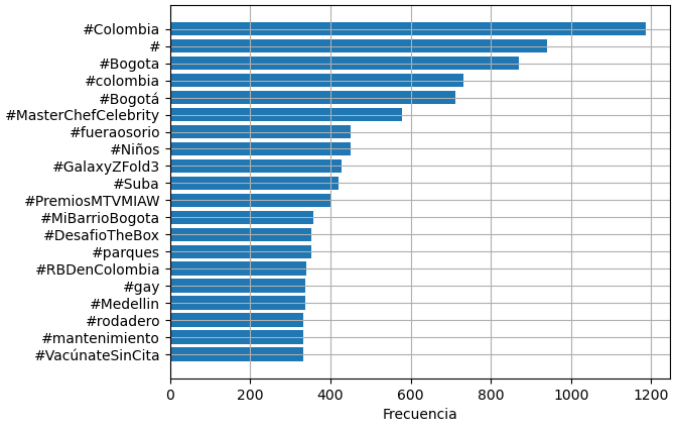


Figura 15: hashtags más comunes
Fuente: elaboración propia

Ahora, teniendo en cuenta los hashtags mas comunes, es de resaltar que ahora no son de personalidades políticas, sino que resalta el de programas de televisión como MasterChefCelebrity, #PremiosMTVMIAW, o de sucesos deportivos como el descontento manifestado al actual entrenador del deportivo América de Cali, Juan Carlos Osorio, y el hashtag muy popular #fueraosorio.

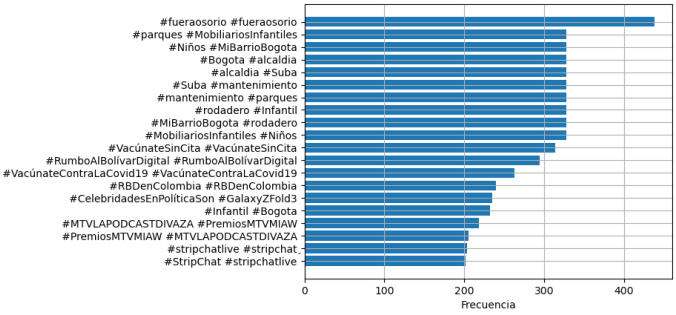


Figura 16: hashtags compuestos más comunes
Fuente: elaboración propia

En cuanto a los hashtags compuestos mas comunes, se puede observar el de #fueraosorio, los de la alcaldía de Bogotá, con #Bogota, #alcaldia, #MiBarrioBogota y también es de resaltar los hashtags que hacen referencia a la vacunación contra el covid 19.

10.1. Ajuste modelo Supervisado Bayes ingenuo

Para mejorar el resultado de los datos, se realizo un test de Chi-Cuadrado, en el cual se tratará cada registro como valor binario, y se determinara si son independientes, solo

manteniendo aquellos los que no se rechazan, con un nivel de significancia del 0.05. Después de esto, se vectorizan los datos, lo que significa que se convierte los datos a una matriz de recuento de tokens. Esta matriz se utilizara en el Pipeline de Scikit-learn, para el entrenamiento del modelo. Se dividen los datos en datos de prueba y de entrenamiento, dejando que los datos de prueba sean el 10 % del total de los datos, y los datos de entrenamiento serán por ende el 90 % restante. Para que los resultados sean reproducibles, se ajusta una semilla o random_state igual a 0.

10.1.1. Evaluación del modelo Bayes Ingenuo

Para la evaluación del modelo entrenado de Bayes ingenuo, se utilizo la base de prueba resultante del 10 % del total de los datos, como se especifico con antelación. Las métricas que se escogieron y sus resultados se presentan en la siguiente tabla: **Accuracy**: 0.76

Polaridad/Métricas	Precisión	Recall	F1 - Score
N	0.79	0.82	0.81
P	0.73	0.68	0.70

Tabla 3: Ajuste modelo Bayes ingenuo

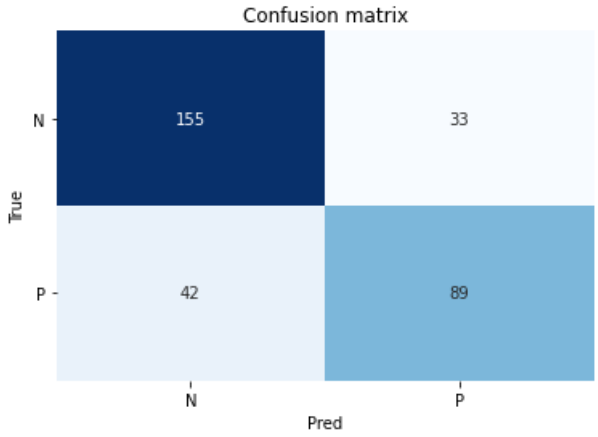


Figura 17: Matriz de confusión modelo bayes ingenuo
Fuente: elaboración propia

En los resultados obtenidos en la tabla superior, así como lo que se puede observar en la matriz de confusión de la Ilustración, el modelo tuvo un desempeño sobresaliente, teniendo un Accuracy de 0.76 en promedio. En las siguientes figuras se puede observar los resultados de la curva ROC y las curvas precisión-recall. **AUC**: 0.84

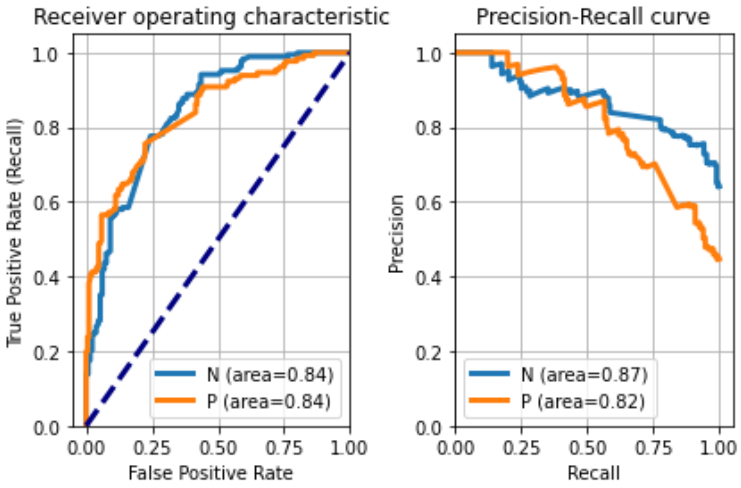


Figura 18: Curvas ROC y Recall modelo bayes ingenuo
Fuente: elaboración propia

Se puede observar en la ilustración que tanto la curva ROC, como las curvas precision-recall, presentan un desempeño sobresaliente, lo que quiere decir que el desbalanceo de los datos no tuvo un efecto sobre el rendimiento del modelo. Se puede observar que se tiene un AUC de 0.84 en promedio. Con los anteriores resultados, se puede tener una gran aproximación al momento de clasificar los tweets que se generan en tiempo real, y que se capturan gracias a la API de Twitter.

10.2. Ajuste del modelo No Supervisado Asignación de Dirichlet Latente

Para ajustar el modelo no supervisado de Asignación de Dirichlet Latente, se crea primero el diccionario de ID's de las palabras ya tratadas, y posteriormente se crea el corpus, como se describió previamente en la metodología. Este corpus y este diccionario se pasan al modelo de LDAMulticore de la librería Gensim, con un parámetro de 10 tópicos. Los resultados del ajuste del modelo se pueden ver a continuación:

Tópicos	Palabras
1	0.007*“hacer” + 0.004*“poder” + 0.004*“confirmar” + 0.004*“decir” + 0.003*“1*“ + 0.003*“petrogustavo*“ + 0.003*“ganar” + 0.003*“bogota*“ + 0.003*“venir” + 0.003*“trabajar”
2	0.016*“bueno” + 0.010*“poder” + 0.007*“hacer” + 0.007*“dia” + 0.006*“decir” + 0.006*“pais*“ + 0.005*“creer” + 0.005*“querer” +0.005*“gobernar” + 0.004*“ver”
3	0.015*“dios*“ + 0.013*“decir” + 0.011*“vida*“ + 0.009*“gracia” + 0.008*“hacer” + 0.006*“ser” + 0.005*“ir” + 0.005*“poder” + 0.005*“tu” + 0.004*“hp*“
4	0.013*“gracia” + 0.011*“hacer” + 0.010*“dar” + 0.008*“mucho” + 0.007*“decir” + 0.006*“dejar” + 0.005*“gente*“ + 0.005*“coser” + 0.005*“saber” + 0.005*“seguir”
5	0.021*“querer” + 0.015*“poder” + 0.007*“hacer” + 0.006*“vida*“ + 0.005*“claudialopez*“ + 0.005*“parecer” + 0.005*“semana*“ + 0.005*“petrogustavo*“ + 0.004*“petro*“ + 0.004*“deber”
6	0.018*“amar” + 0.012*“rico*“ + 0.011*“hacer” + 0.008*“ganar” + 0.007*“pasar”+ 0.007*“querer” + 0.006*“hermoso” + 0.005*“necesitar” + 0.005*“poder” + 0.005*“ay*“
7	0.033*“acabar” + 0.027*“foto*“ + 0.025*“publicar*“ + 0.024*“colombia*“ + 0.009*“risa“ + 0.009*“bogota“ + 0.007*“antioquia*“ + 0.006*“santo” + 0.006*“video*“ + 0.005*“gustar”
8	0.112*“*“ + 0.009*“amor*“ + 0.009*“ir” + 0.007*“ver” + 0.007*“volver” + 0.006*“querer” + 0.005*“llegar” + 0.004*“odiar” + 0.004*“dosis*“ + 0.004*“hablar”
9	0.029*“ir” + 0.010*“hacer” + 0.009*“querer” + 0.007*“decir” + 0.007*“salir” + 0.006*“personar” + 0.006*“vida*“ + 0.005*“todo” + 0.005*“ver*“ + 0.005*“creer”
10	0.013*“hacer” + 0.009*“contar” + 0.006*“hola*“ + 0.006*“partir” + 0.006*“decir” + 0.006*“dar” + 0.005*“total*“ + 0.005*“ir” + 0.004*“medellin*“ + 0.004*“pues*“

Tabla 4: Resultados modelo Asignación de Dirichlet Latente

¿Cómo se leen los resultados? El tópico 1 está representado por 0.007*“hacer” + 0.004*“poder” + 0.004*“confirmar” + 0.004*“decir” + 0.003*“1*“ + 0.003*“petrogustavo*“ + 0.003*“ganar” + 0.003*“bogotá*“ + 0.003*“venir” + 0.003*“trabajar”.

Esto quiere decir que las palabras principales del tópico 1 son «hacer, poder, confirmar, decir, 1, petrogustavo, ganar, Bogotá, venir y trabajar». El peso de la palabra “hacer” es

de 0.007. Podríamos ponerle un titulo a este tópico como: “Petro Ganador”, pues se puede inferior que tiene gran referencia a que el senador Gustavo Petro gane las elecciones, y en Bogotá lo apoyan. Así sucesivamente se puede comenzar a analizar los tópicos.

10.2.1. Evaluación del modelo Asignación de Dirichlet Latente

Pero, ¿Cómo se sabe cuántos tópicos son los correctos? Se realizo una iteración para saber cuántos tópicos son los más apropiados, utilizando como métrica de evaluación la de Topic Coherence UMass.

Calcula la frecuencia con la que dos palabras, w_i y w_j aparecen juntas en el corpus y se define como (Zvornicanin, 2021)

$$C_{UMass}(w_i, w_j) = log\frac{D(W_i, w_j) - 1}{D(w_i)} \tag{21}$$

Zvornicanin (2021) describía la anterior ecuación de la siguiente manera: *”$D(w_i, w_j)$ indica cuántas veces las palabras w_i y w_j aparecen juntas en los documentos, y $D(w_i)$ es cuántas veces apareció la palabra w_i sola. Cuanto mayor sea el número, mejor será la puntuación de coherencia. Además, esta medida no es simétrica, lo que significa que $C_{UMass}(w_i, w_j)$ no es igual a $C_{UMass}(w_j, w_i)$. Se calcula la coherencia global del tema como las puntuaciones medias de coherencia por pares en las N palabras principales que describen el tema”.*

La grafica con los resultados se presenta a continuación.

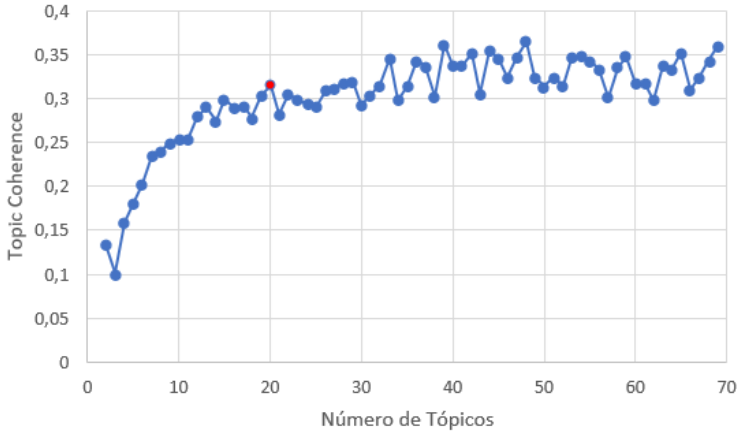


Figura 19: Topic Coherence modelo Asignación de Dirichlet Latente
Fuente: elaboración propia

Se Puede observar que, en 20 tópicos, es un resultado aceptable, pues hacer más de 29 tópicos comienza a complicar la interpretación y observación de los mismos.

No hay una manera de determinar si la puntuación de la métrica es buena o mala, pues esto depende de los datos y el objetivo, además de que existe una gran cantidad

de métricas alternativas,las cuales destacan la distancia de Jaccard, Coherencia UCI, Coherencia CV, y y otras. El usuario es el que debe elegir al final con cual quedarse, y tomar como guía el resultado de las métricas (Zvornicanin, 2021). Usualmente la métrica se vuelve más alta con más tópicos, pero el incremento cada vez sera menor, por lo cual una gráfica de codo puede ser útil, como la gráfica de la figura 19.

Se puede observar los resultados con 20 tópicos en la siguiente tabla:

Tópicos	Palabras
1	0.013*“encantar” + 0.008*“dudar” + 0.008*“twitter*“ + 0.008*“hermoso*“ + 0.008*“mundo*“ + 0.007*“rico*“ + 0.007*“querer” + 0.006*“pereira*“ + 0.006*“dar” + 0.005*“bueno”
2	0.015*“ganar” + 0.009*“ivanduque*“ + 0.009*“ir” + 0.007*“colombia*“ + 0.006*“poner” + 0.006*“minsaludcol*“ + 0.005*“hpta*“ + 0.005*“salir” + 0.005*“x2*“ + 0.005*“colombia*“
3	0.015*“bueno” + 0.008*“perder” + 0.008*“hacer” + 0.007*“mierda*“ + 0.007*“colombia*“ + 0.006*“ex” + 0.006*“durar” + 0.006*“alvarouribevel*“ + 0.005*“matar*“ + 0.005*“poder”
4	0.021*“mariafdacabal*” + 0.016*“decir” + 0.014*“jovenes” + 0.013*“alguien*“ + 0.012*“gustar” + 0.011*“poder” + 0.010*“perder” + 0.009*“pasar” + 0.009*“cabal*“ + 0.009*“pensar”
5	0.007*“petro*“ + 0.007*“todo” + 0.007*“ganar” + 0.006*“presidente*“ + 0.006*“encuestar” + 0.006*“querer” + 0.005*“decir” + 0.005*“super*“ + 0.005*“colombia*“ + 0.004*“felicidad*“
6	0.013*“ir” + 0.013*“necesitar” + 0.013*“paso” + 0.008*“ver*“ + 0.008*“mar” + 0.007*“cartagena*“ + 0.007*“confirmar” + 0.006*“poder” + 0.006*“verdad*“ + 0.005*“viajar*“
7	0.029*“contagio” + 0.016*“decir” + 0.015*“viajero” + 0.010*“ser” + 0.008*“total*“ + 0.006*“vacunar” + 0.006*“bueno*“ + 0.005*“señor*“ + 0.005*“querer” + 0.005*“covid*“
8	0.015*“santo” + 0.008*“iglesia” + 0.008*“amigo” + 0.007*“ir” + 0.007*“julio*“ + 0.007*“decir” + 0.006*“bucaramanga*“ + 0.006*“rt*“ + 0.006*“hacer” + 0.005*“danieles*“
9	0.024*“dia” + 0.016*“hola*“ + 0.011*“bueno” + 0.009*“covid*“ + 0.007*“dosis*“ + 0.007*“muerte*“ + 0.007*“segundar” + 0.007*“ir” + 0.006*“uci” + 0.006*“vez*“
10	0.014*“hermoso” + 0.008*“amor*“ + 0.008*“cali*“ + 0.008*“gente*“ + 0.007*“poder” + 0.007*“jorgeivanospina*“ + 0.006*“odiar” + 0.005*“gonorrea*“ + 0.005*“primera” + 0.004*“linea”

Tópicos	Palabras
11	0.019*“ver” + 0.012*“venezuela” + 0.009*“extrañar” + 0.008*“llegar” + 0.007*“veneco” + 0.006*“llamar” + 0.006*“sentir” + 0.005*“cucuta*“ + 0.005*“barrancabermeja*“ + 0.005*“frontera”
12	0.013*“brasil” + 0.013*“partido” + 0.012*“jugar” + 0.011*“copa” + 0.010*“poder” + 0.007*“america” + 0.007*“hablar” + 0.007*“falto” + 0.007*“messi” + 0.005*“soñar”
13	0.021*“marchar” + 0.009*“inclusion” + 0.008*“dar” + 0.008*“trabajar” + 0.007*“seguir” + 0.007*“vacuno” + 0.005*“orgullo” + 0.005*“saber” + 0.005*“gente*“ + 0.004*“gay*“
14	0.199*“bogota“ + 0.013*“cali*“ + 0.006*“ir” + 0.006*“querer” + 0.006*“claudialopez*“ + 0.006*“manizales*“ + 0.006*“medellin*“ + 0.005*“hacer” + 0.004*“poder” + 0.004*“personar”
15	0.017*“poder” + 0.010*“colombia*“ + 0.010*“hacer” + 0.009*“gustavobolivar*“ + 0.009*“querer” + 0.008*“pacto” + 0.006*“alcalde*“ + 0.005*“pais*“ + 0.005*“tetas” + 0.005*“petrogustavo*“
16	0.016*“risa“ + 0.007*“valer” + 0.006*“mascota” + 0.006*“querer” + 0.005*“dar” + 0.005*“perro*“ + 0.005*“hacer” + 0.005*“poder” + 0.004*“andar” + 0.004*“ir”
17	0.070*“acabar” + 0.058*“foto*“ + 0.055*“publicar*“ + 0.038*“colombia*“ + 0.021*“bogota*“ + 0.016*“antioquia*“ + 0.013*“video*“ + 0.009*“medellin*“ + 0.008*“santander*“ + 0.007*“ir”
18	0.022*“cambiar“ + 0.010*“bueno” + 0.010*“pene“ + 0.009*“petrogustavo*“ + 0.007*“serio” + 0.007*“uff*“ + 0.007*“votar” + 0.007*“decimetro” + 0.006*“bello” + 0.006*“tarde”
19	0.034*“vida*“ + 0.025*“dios*“ + 0.019*“feliz*“ + 0.016*“gracia” + 0.012*“hacer” + 0.010*“mucho” + 0.010*“saber” + 0.009*“querer” + 0.009*“personar” + 0.009*“coser”
20	0.039*“amar” + 0.018*“iglesia” + 0.011*“comer” + 0.009*“volver” + 0.008*“lindo” + 0.006*“pareja” + 0.006*“creer” + 0.006*“sexo” + 0.006*“parecer” + 0.006*“casar”

Tabla 5: Tópicos resultado Asignación de Dirichlet Latente

Se pude observar que, en el primer tópico, se tienen palabras positivas sobre el mundo, y Pereira, posiblemente en un contexto turístico. Es interesante observar que en los

tópicos 2 y 3 se tienen palabras despectivas con respecto a ivanduque, que es el perfil de Twitter del presidente Ivan Duque, y con alvarouribevel, que es el perfil de Twitter del ex presidente Álvaro Uribe Vélez. En el tópico 5, se puede observar que se tiene la palabra petro, haciendo referencia al apellido del senador Gustavo Petro, y se tiene la palabra ganar, presidente y felicidad, lo que hace referencia que para muchas personas, seria de gran felicidad que el senador ganara las elecciones presidenciales el próximo año.

Ahora bien, teniendo en cuenta el modelo que se construyo de Bayes Ingenuo, se puede aplicar para saber la proporción de Tweets que son positivos o negativos. Antes de aplicarlo, se debe tener en cuenta que el modelo de Bayes Ingenuo fue entrenado sobre datos a los que se le hicieron stemming, por lo cual, para aplicarlo a los datos recopilados mediante la API de Twitter, es menester realizar el mismo proceso antes de clasificar con el modelo. Se obtuvieron los siguientes resultados:

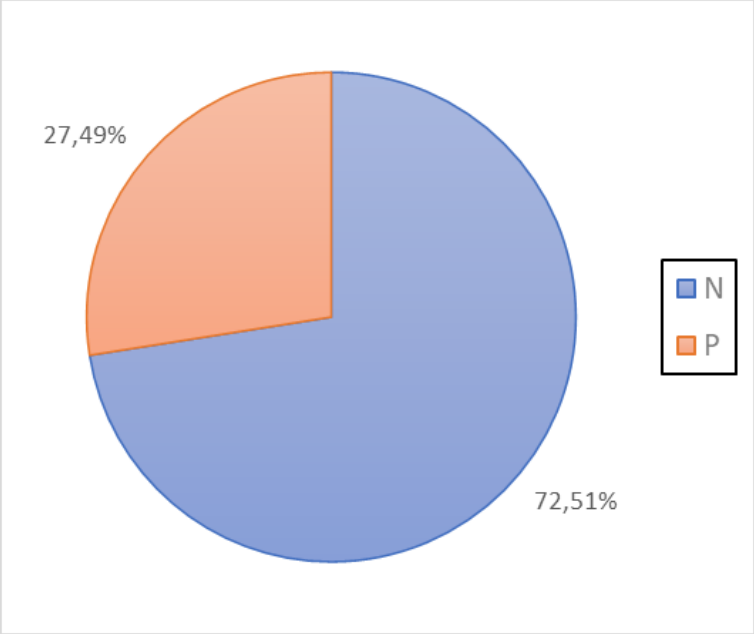


Figura 20: Clasificación Polaridad Modelo
Fuente: elaboración propia

Se puede observar en la gráfica, el resultado proporcional del numero de Tweets que son catalogados como positivos, y otros como negativos. Se puede observar que, del total de tweets recolectados, el modelo clasifico el 27.49 % como tweets positivos, y el 72.51 % como tweets negativos. Algunos ejemplos de tweets se pueden observar a continuación:

Posición	Contenido	Clasif.
11	el quindío que hoy celebra sus 55 años como departamento ha demostrado ser tierra acogedora dueña de hermosos paisajes y cuna de cafetales que conforman inolvidables paisajes atractivos que las agenciasdeviajes recomiendan sin faltapaulacortesc quindiogob	P
37	felicidad descripción grafica	P
157307	el amor es una fuerza motivadora que nos mantiene en pie a pesar de la adversidad el amor a dios el amor a	P
87745	gracias dona_pily2 por siempre ayudar	P
275208	me siento muy orgulloso de proponerme un objetivo y poderlo cumplir gracias al apoyo de mi familia y la dedicación puede que para algunos sea último modelo ni mucho menos gama alta pero para mí es una gran satisfacción porque es el fruto y recompensa de mi esfuerzo	P
342	wradiocolombia q maravilla se reactiva la economía	P
297137	recomendado noticias día internacional de la niña celebrar con las niñas para construir con las niñas el 11 de octubre se celebra el día internacional de la niña una fecha acordada globalmente lo nuevo hoy	P

Tabla 6: Ejemplo tweets catalogados positivos

Se puede observar en la tabla, un ejemplo de tweets que fueron catalogados por el modelo como tweets positivos, y en gran medida el modelo esta haciendo bien su trabajo. El modelo reconoce con eficacia palabras como felicidad, amor, recomendar, maravilla y demás palabras positivas.

Posición	Contenido	Clasif.
60	en lo que va del año en total han sido reportadas 41 masacres en 35 municipios del país en este mapa presentamos las regiones en las que se han presentado estos hechos violentos a lo largo del 2021 navega el mapa aquísigue hilo	N
348105	que dolor de cordales tan hpta	N
45373	showdelasestrellas canalrcn años han pasado desde que vi un programa de rcn hoy por casualidad veo el show de las estrellas y ver el ridículo que le hacen hacer a un caballero y personaje ilustre y respetable como el sr jorgebarontv me parece denigrante	N
71584	alvarouribevel juanmansantos es el mismisimo demonio en persona vendió el país y es el papá de la guerrilla	N
161	detesto ver mi escritorio desordenado	N
9273	todo este poco de hps les importa todo menos las regiones y su gente	N

Tabla 7: Ejemplo tweets catalogados negativos

En cuanto a la anterior tabla, se observa que el modelo también es eficiente en clasificar tweets negativos, en la categoría de negativos. Es capaz de identificar palabras como masacre, muerte, ridiculez, groserías y demás palabras negativas. Pero el modelo, aunque clasifica muy bien en la mayoría de los casos, en la mayoría de registros, como todo en la vida, no es perfecto, y tiene algunas falencias que se deben pulir.

Posición	Contenido	Clasif.
371310	me dijo que el clima estaba rico y mira donde acabamos	P
55	indigenamuisca	N
74	en blanco y negrochelog	N
61	en el último mes los departamentos más afectados por este tipo de delitos han sido antioquia cauca caquetá y cesar conozca aquí los principales responsables y victimas en estos territorios navega el mapa de masacres aquí cerramos hilo	P
76	sophiefajardoo cual	N
307	dios mío	P
166	vieron está belleza	N
203606	bravooooo ganó liss ay qué felicidámasterchefcelebrity lisspereira	N
182	entonces el futuro está en el monopolio	P
228393	gustavobolivar peñalosa cuando fracasóhay están las obras que ustedes están ayudando a destruir no hable m	P

Tabla 8: Ejemplo tweets mal catalogados

Se puede observar en la anterior tabla algunos errores de mal interpretación del modelo, y mala clasificación. Muchas veces las palabras positivas como ayudar, se utilizan en contextos negativos como “ayudar a destruir”, con lo cual el modelo sufre de una confusión, y erra en el momento de clasificar. Muchas otras veces, los usuarios publican tan solo una palabra o expresión, que es muy poco conocida o incluso sin sentido, a lo cual el modelo lo clasifica como una categoría negativa. Errores que son pulibles con el tiempo y el desarrollo del modelo.

11. Discusión y Conclusiones.

11.1. Conclusiones del Modelo.

El modelo demuestra ser suficiente y eficiente para categorizar las palabras, por su co ocurrencia. Esto quiere decir, el modelo al elegir los tópicos, es influenciado por aquellas palabras que aparecen junto con otras (co ocurrencia). Por ejemplo, la palabra playa aparece muchas con la palabra mar; la palabra pistola aparece muchas veces junto con la palabra bala, y así sucesivamente.

El modelo no tiene una métrica exacta que permita determinar el numero optimo de tópicos, por lo cual, se desarrollaron técnicas alternativas con el tiempo, por parte de autores ajenos al Desarrollo del modelo (Umass, CV, UCI, Word2Vec, Jaccard). Lo anterior, es un punto de oportunidad, en el cual se puede construir un modelo mejor, que tenga en cuenta esta parte, o crear una extensión que mejore este aspecto del modelo LDA.

Las probabilidades de las palabras son maximizadas, al dividirse el numero de estas por el numero de tópicos. Por ejemplo, mas palabras significa mas información para ser localizada en el simplex del triangulo. Entre más información se tenga, y a información se hace referencia a palabras, se pueden obtener tópicos con mejor significado para el usuario; esto también esta influenciado por la calidad de la depuración que se realice al conjunto de documentos.

En LDA no importa que un documento este compuesto por varios tópicos, pues la distribución de Dirichlet lo obliga a pertenecer al extremo con mayor probabilidad. Pero esto esta altamente influenciado por la elección de los hiper parámetros α y η.

11.2. Conclusiones de la Aplicación.

El producto desarrollado, ha sido completamente desarrollado con herramientas de código abierto, que están disponibles para su libre descarga en la web. A través de esta herramienta, se ha podido obtener una vista rápida sobre lo que piensan los usuarios de Twitter en las ciudades capitales de Colombia. Ajustando dos modelos (uno de aprendizaje supervisado y otro de aprendizaje no supervisado) se logró identificar superficialmente dos características importantes: la polaridad que esconden los comentarios escritos en los tweets, gracias al modelo de Bayes Ingenuo; y cuáles son las temáticas principales que se están tratando, gracias al modelo de Asignación de Dirichlet Latente.

Es de resaltar, que el filtro que se realizó para la recolección de datos utilizando la API de Twitter, y la librería de Python: tweepy, fue de tan solo la ubicación en la que se realiza la captura de los datos, utilizando la herramienta de la web https://boundingbox.klokantech.com/. La polaridad de los tweets extraídos de los datos de TASS, permitió observar que la mayoría de los tweets de los usuarios, de diferentes países de habla hispana, son de polaridad negativa. Se puede decir que la mayoría de los usuarios acuden a Twitter para quejarse sobre lo que está sucediendo en su entorno, que ha agradecer o alabar las cosas que suceden.

Con el entrenamiento del modelo clasificatorio, gracias a estos datos ya etiquetados, se logró clasificar de igual manera los datos obtenidos a través de la API de Twitter, y se obtuvieron resultados similares. El 73 % de los usuarios de Twitter en las principales ciudades capitales de Colombia, acuden a la red social para publicar micro blogs negati-

vos, y el 27 % publican mensajes positivos. En pocas palabras, se puede afirmar que los usuarios usan la red social Twitter para quejarse de la coyuntura actual de su entorno, mas que para agradecer, o publicar contenido aleatorio; la política es sobre lo que mas se quejan los usuarios en las ciudades principales de Colombia. Esta herramienta reduce los costos de realizar encuestas, puesto que se elimina la necesidad de intermediarios, además de la inmediates de los datos al ser en tiempo real la recolección de los datos.

En términos políticos, las personas están divididas principalmente en temas de política neoliberal o de política socialista, y en Colombia, los principales representantes de estos temas se pueden considerar el senador Gustavo Petro, con afinidades de izquierda, el senador Gustavo Bolívar que de igual forma tiene una tendencia de pensamiento político de izquierda, y por pensamientos políticos de derecha se tienen al actual presidente Ivan Duque, y al ex presidente Álvaro Uribe Vélez.

Se observo que en temas que tienen que ver con el senador Gustavo Petro, se relaciona con palabras positivas como ganar, presidente, gobernar, mientras que temas que tienen que ver con personalidades como el ex Presidente Uribe Vélez, el presidente Ivan Duque o al ministerio de salud, se relaciona más con palabras negativas y despectivas. No son extraños estos resultados, sino que antes es confirmación de la mala percepción que tienen los usuarios de Twitter colombianos de las principales ciudades colombianas con relación a sus actuales lideres políticos. También se confirma la buena percepción política que ostenta actualmente el senador alternativo Gustavo Petro. Sin haber realizado un estudio exhaustivo de encuestas de satisfacción política, se puede afirmar que los usuarios se sienten totalmente inconformes con los actuales lideres políticos que tienen el poder en Colombia, y desean un cambio en de gobierno, siendo el principal representante de este cambio el senador Petro.

Se observa también que los colombianos tienen en mente el proceso de vacunación contra la covid 19, siendo uno de las principales arrobas que se utilizan. Dato que es especial teniendo en cuenta la cantidad de casos fatales de covid 19, y además de esto, el impacto que tuvo en la economía nacional debido a las cuarentenas obligatorias que se tuvieron que decretar a nivel nacional.

También se encontró que de los principales hashtags que se tienen, se observa el de programas televisivos y eventos como los de Masterchef Celebrity, y los premios de MTV. No solo de programas de televisión se puede ver, sino de productos que llegaron a irrumpir en el mercado como lo fue el novedoso teléfono celular Samsung Z Fold 3, que ha tomado gran acogida y admiración debido a su diseño plegable novedoso.

Se puede decir que el nuevo producto de la marca de celulares Samsung, ha tomado una relevancia especial por su innovación, por sus nuevas funcionalidades, y se ha posicionado incluso por delante de otras marcas icónicas como lo es Apple, Huawei y demás.

No solo esta herramienta funciona para saber la opinión de los usuarios sobre los temas políticos vigentes, sino sobre temas populares como programas televisivos, sobre actores, o productos populares, como por ejemplo, teléfonos celulares. Con los resultados de las opiniones de los usuarios de la red social, se pueden tomar decisiones bastante importantes en cuanto a percepción de las personas, para comenzar a incrementar la buena imagen en los usuarios y personas en general sobre determinado candidato político, programa televisivo, persona influyente, famosa o altamente reconocida, así como imagen de marca empresarial.

En el actual trabajo realizado, hubieron partes que quedaron fuera del proyecto debido

a la generalidad del mismo; pero el actual proyecto se puede aplicar de una forma mucho más personalizada, abarcando una temática de estudio mucha más específica. En el actual estudio se abarco las ciudades capitales de Colombia, pero se podría abarcar por temáticas políticas que contengan la palabra JEP, Reforma, Cadena Perpetua, o por ejemplo se puede filtrar por el nombre se partidos políticos como Centro Democrático, Cambio Radical, etc. Además de esto, se puede filtrar para que el código realice un stream de datos por cuentas de Twitter en específico, y de esta manera solo obtener datos de personalidades en especifico o de paginas de Twitter pertenecientes a empresas, entidades, ONG's, etc.

Por otro lado, los algoritmos utilizados en el actual trabajo se podrían complementar con procedimientos adicionales que pueden mejorar su rendimiento. En el caso del algoritmo de análisis semántico, se puede complementar con diferentes técnicas de modelado de tópicos como "Word embeddings". En el caso del algoritmo de Bayes Ingenuo, al estudio de polaridad se puede sumar mas tipos de polaridades, y en este caso entraría directamente el sentimiento neutral, pero para el análisis de esto se debe tener en cuenta la implementación de técnicas para modelado de datos no balanceados, la aplicación de métodos de penalización, o cambiar el algoritmo por otro algoritmo de boosting.

Con estas nuevas implementaciones, se podría comenzar a pensar en la parte de la implementación, de utilizar un Hardware con mayor capacidad y poder de procesamiento, que permita realizar el procesamiento de muchos más datos, millones de datos, sin tener que esperar varias horas para arrojar el resultado de los modelos, sino que el resultado pueda ser resuelto en cuestión de minutos o incluso segundos, creando así una herramienta más inmediata en tiempo real. Junto con esto, se puede evaluar la posibilidad de la implementación de esta herramienta en un ambiente de producción, muy probablemente haciendo uso de herramientas de la nube, proporcionadas por Azure, AWS, Google Cloud, por poner un ejemplo de las más famosas.

12. Referencias, Inspiración y Bibliografía.

Referencias

Bengfort, B., Bilbro, R. & Ojeda, T. (2018), *Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning*, O'Reilly Media, Inc.

Bécue Mónica, L. L. (1992), 'El analisis estadistico de datos textuales. La lectura según los escolares de enseñanza primaria.', *Anuario de Psicología. Universitat de Barcelona.* .

Golberg, Y. (2017), *Neural Network Methods in Natural Language Processing*, Morgan & Claypool Publishers.

Guarnizo, E. & Monroy, A. (2020), Implementación de un modelo de Análisis de sentimientos con respecto a la JEP basado en minería de datos en twitter, Master's thesis, Universidad Católica de Colombia, Colombia.

Hammoe, L. (2018), Detección de Tópicos utilizando el modelo LDA, Master's thesis, Instituto Tecnológico de Buenos Aires, Argentina.

Hernández, H. (2020), Integración de Data Mining sobre noticias para predicción de mercados financieros, Master's thesis, Universidad Politécnica de Madrid, España.

Ingersoll, G., Morton, T. & Farris, D. (2013), *Taming Text: How to Find, Organize, and Manipulate It*, Manning Publications.

Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y. & Zhao, L. (2019), 'Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey', *Multimed Tools Appl* **78**.

Kapadia, S. (2019), 'Topic Modeling in Python: Latent Dirichlet Allocation (LDA)', *towards data science* .

Kontostathis, A. (2007), 'Essential Dimensions of Latent Semantic Indexing (LSI)', *Department of Mathematics and Computer Science. Ursinus College* .

Lane, H., Hapke, H. & Howard, C. (2019), *Natural Language Processing in Action: Understanding, analyzing, and generating text with Python*, Manning Publications.

Manning, C. D. & Schütze, H. (1999), *Foundations of Statistical Natural Language Processing*, MIT Press.

Arley Rincon, W. (2014). Preguntas abiertas en encuestas¿ cómo realizar su análisis? Bogotá.

Griffiths, T. (2004). Gibbs Sampling in the Generative Model of Latent Dirichlet Allocation. USA.

Griffiths, T. a. (2004). Finding Scientific Topics. USA: Proceedings of the National Academy of Sciences of the United States of America.

kedarps. (2019). https://stats.stackexchange.com. Retrieved from https://stats.stackexchange.com/users/148774/kedarps

Zvornicanin, E. (2021). When Coherence Score is Good or Bad in Topic Modeling? Los Angeles.

Pritchard JK, S. M. (2000). Inference of population structure using multilocus genotype data. In Genetics 155 (pp. 945-959)

0Barrios Arce, J. (26 de julio de 2019). www.juanbarrios.com. Obtenido de https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

Davis, J. (2006). The Relationship Between Precision-Recall and ROC Curves. Madison, WI, USA.

Geigle, C. (2017). Inference Methods for Latent Dirichlet Allocation. Illinois.

Ponweiser, M. (2012). Latent Dirichlet Allocation in R. Vienna.

Speh , J., Muhic , A., Rupnik, J. (2021). Parameter Estimation for the Latent Dirichlet Allocation. Eslovenia.

Roder, M., Both, A., Rosner, F., Et.Al. (2014). Evaluating topic coherence measures. Cornell University.

AFP. (25 de mayo de 2021). El paro influyó en imagen desfavorable de Duque y la Policía. Portafolio.

Ahlgren, M. (18 de agosto de 2021). www.websiterating.com. Obtenido de Más de 50 estadísticas y datos de Twitter: https://www.websiterating.com/es/research/twitter-statistics/

Alvino, C. (12 de abril de 2021). Estadísticas de la situación digital de Colombia en el 2020-2021. Obtenido de branch.com.co: https://branch.com.co/marketing-digital/estadisticas-de-la-situacion-digital-de-colombia-en-el-2020-2021/

Haselmayer, M., Jenny, M. (2016). Sentiment analysis of political communication: combining a dictionary approach with crowdcoding. Springer.

Mainou, R. G. (13 de agosto de 2019). Autoestima, validación y redes sociales. El Economista.

valora analitik. (30 de abril de 2021). Desempleo Colombia: marzo con la menor tasa desde la pandemia. Obtenido de https://www.valoraanalitik.com/2021/04/30/desempleo-colombia-con-la-menor-tasa-desde-la-pandemia/

Besancon, R., Rajman, M. (2013). Text Mining: Natural Language techniques and Text Mining applications. Swiss Federal Institute of Technology.

Blei, D. M., Ng, A. Y., Jordan, M. I. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 993-1022.

Dave, M., Sharma, V. (2012). SQL and NoSQL Databases. International Journal of Advanced Research in Computer Science and Software Engineering.

Hart, P., Duda, R. (1973). Pattern Classification and Scene Analysis. John Wiley and Sons.

Kontostathis, A. (2004). Essential Dimensions of Latent Semantic Indexing (LSI). Ursinus College.

Lin, J. (2016). On The Dirichlet Distribution. Queen's University.

van Zyl , C. (2018). Frequentist and Bayesian inference: A conceptual primer. New Ideas in Psychology, 44-49.

Vázquez Marcos, J. (2017). Modelado de Tópicos para perfilado de Blogs. Madrid: UNIVERSIDAD CARLOS III DE MADRID.

A. Códigos

A.1. Código LDA.

```
# ========================================================================
# Librerias a Ser Usadas
# ========================================================================

import pandas as pd
import numpy as np
import glob
import os
import re
import nltk
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize
from nltk.corpus import stopwords
import string
import matplotlib.pyplot as plt
import smart_open
smart_open.open = smart_open.smart_open
import gensim
from gensim.models import CoherenceModel
from gensim.corpora import Dictionary
import spacy
from spacy.lang.es.examples import sentences
import es_core_news_sm
import gensim.corpora as corpora
from gensim.models.ldamulticore import LdaMulticore
from pprint import pprint



from nltk.corpus import words


spanish_stopwords = stopwords.words('spanish')
english_stopwords = stopwords.words('english')


nlp = es_core_news_sm.load()


#%%

# ========================================================================
# Importacion de los Datos.
```

```
# ============================================================================

path = r'C:\Users\manuel.diaz\Documents\T' # use your path

all_files = glob.glob(path + "/*.csv")

li = []

for filename in all_files:
    df = pd.read_csv(filename, index_col=None, header=0)
    li.append(df)

frame = pd.concat(li, axis=0, ignore_index=True)

#%%

# ============================================================================
# Creacion Funcion de Lematización
# ============================================================================

#encoding: utf8
# Se importa la base de datos lemmatization-es.txt que contiene
# mas de 200.000 palabras en español
lemmaDict = {}
with open(r'C:\Users\manuel.diaz\Downloads\lemmatization-es.txt', 'rb') as f:
   data = f.read().decode('utf8').replace(u'\r', u'').split(u'\n')
   data = [a.split(u'\t') for a in data]



for a in data:
   if len(a) >1:
      lemmaDict[a[1]] = a[0]

def lemmatize(word):
   return lemmaDict.get(word, word + u'*')


#%%

# ============================================================================
# Creacion Funcion para la Limpieza de datos.
# ============================================================================

def cleaner(word):

    # listas vacias
    list_word_clean1 = []
    list_word_clean2 = []
    lemma = []
```

```
original = str(word)
word = re.sub(r'((http|https)\:\/\/)?[a-zA-Z0-9\.\/\?\:@\-_=#]+\.([a-zA-Z]){2,6}([a-zA-Z0-9\.\&\/\?\:@\-_=#])*'
word = re.sub(r'(?::|;|=)(?:-)?(?:\)|\(|D|P)', "", str(word))
word = re.sub(r'ee.uu', 'eeuu', str(word))
word = re.sub(r'\#\.', '', str(word))
word = re.sub(r'\n', '', str(word))
word = re.sub(r',', '', str(word))
word = re.sub(r'\-', ' ', str(word))
word = re.sub(r'\.{3}', ' ', str(word))
word = re.sub(r'a{2,}', 'a', str(word))
word = re.sub(r'é{2,}', 'é', str(word))
word = re.sub(r'i{2,}', 'i', str(word))
word = re.sub(r'ja{2,}', 'ja', str(word))

# word = re.sub(r'á', 'a', word)
# word = re.sub(r'é', 'e', word)
# word = re.sub(r'í', 'i', word)
# word = re.sub(r'ó', 'o', word)
# word = re.sub(r'ú', 'u', word)
# word = re.sub('[^a-zA-Z]', ' ', word)
word = re.sub(r'[^\w\s]', '', str(word).lower().strip())
word = re.sub(r'(a{0,2}j{1,2}a{0,2}){2,}j?', '', str(word))
# word = re.sub(r'\b(?:e{0,2}j{1,2}e{0,2}){2,}j?\b', "risa", str(word))
# word = re.sub(r'\b(?:i{0,2}j{1,2}i{0,2}){2,}j?\b', "risa", str(word))
# word = re.sub(r'\b(?:o{0,2}j{1,2}o{0,2}){2,}j?\b', "risa", str(word))
word = re.sub(r'\b(?:ua{0,2}j{1,2}ua{0,2}){2,}j?\b', "risa", str(word))
# word = re.sub(r'\b(?:o{0,2}l{1,2}o{0,2}){2,}l?\b', "risa", str(word))
word  =  re.sub(' +', ' ', str(word))
word  = word.strip()
word  =  re.sub('\r', '', str(word))



for w1 in word.split(" "):
  if  w1.lower() not in spanish_stopwords:
    list_word_clean1.append(w1.lower())



for w2 in list_word_clean1:
  if  w2.lower() not in english_stopwords:
    list_word_clean2.append(w2.lower())

out_text = " ".join(list_word_clean2)


for a in list_word_clean2:
    if (a not in spanish_stopwords) and (a not in english_stopwords):
        lemma.append(lemmatize(a))
```

```
    at = re.findall(r'[@]\S*', str(original))
    hashtags = re.findall(r'[#]\S*', str(original))

    return word, list_word_clean2,out_text , at,hashtags, lemma

#%%

# ============================================================================
# Aplicacion de las Funciones de lematizacion y limpieza
# a los datos importados en formato CSV.
# ============================================================================
import time

# Contabilizar tiempo
start = time.time()

#listas vacias
list_untoken = []
list_tokenize = []
list_tokenize_united = []
word_lemma = []
hashtags = []
at = []

df = frame.copy()

# ajuste de los datos y guardado en las listas vacias
for i in frame["Text"]:
    a,b,c,d,e, f =  cleaner(word = i)
    list_untoken.append(a)
    list_tokenize.append(b)
    list_tokenize_united.append(c)
    word_lemma.append(f)
    at.append(d)
    hashtags.append(e)

# agregar los datos al dataframe creado.
df["list_untoken"] = list_untoken
df["list_tokenize"] = list_tokenize
df["word_lemma"] = word_lemma
df["at"] = at
df["hashtags2"] = hashtags

end = time.time()
print(end - start)

# tiempo promedio: 90 segundos

# guardar los datos en formato PKL
```

```
df.to_pickle("prueba.pkl")


#%%

# ============================================================================
# Creacion del Diccionario de Frecuencias y el Corpus.
# ============================================================================


id2word = corpora.Dictionary(df['word_lemma'])
# Crear Corpus

texts = df['word_lemma'].copy()
# Documento de Frecuencia de Terminos

corpus = [id2word.doc2bow(text) for text in texts]
# Observar algunos datos

print(corpus[1:2][0][:30])

#%%

# ============================================================================
# Ajuste del Corpus y el Diccionario de Frecuencias
# al modelo LDA.
# ============================================================================

# numero de topicos
num_topics = 10

# Construir modelo LDA
lda_model = LdaMulticore(corpus=corpus,id2word=id2word,num_topics=num_topics,
                         random_state=124) # se ajusta una semilla a los datos

# Imprimir los 10 primeros Temas (10 por defecto)
pprint(lda_model.print_topics())


#%%

# ============================================================================
# Se evalua el Modelo Utilizando las Métricas de
# Coherencia de Topicos.
# ============================================================================

coherence_model_lda = CoherenceModel(model=lda_model, texts=df3['word_lemma'],
                                     dictionary=id2word, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('Coherence Score: ', coherence_lda) # imprimir la medida de Coherencia
```

```
#%%

# ============================================================================
# Funcion para Iterar y Ajustar el Modelo LDA con diferente
# Numero de Topicos.
# ============================================================================

def compute_coherence_values(corpus, dictionary, k):

    lda_model = gensim.models.LdaMulticore(corpus=corpus, # corpus
                                           id2word=id2word, # diccionario
                                           num_topics=k,  # numero de topicos
                                           random_state=124) # semilla

    coherence_model_lda = CoherenceModel(model=lda_model, texts=df3['word_lemma'],
                                         dictionary=id2word, coherence='c_v')

    return coherence_model_lda.get_coherence()
#%%

# ============================================================================
# Iterar e ir Cambiando el numero de topicos, mientras se ajusta el modelo LDA,
# y ademas de esto, ir guardando la metrica de Coherencia de Topicos
# para encontrar la mejor medida.
# ============================================================================

import tqdm #libreria que permite observar el progreso del proceso

grid = {} #grilla
grid['Validation_Set'] = {} # set de validacion

# rango de topicos
min_topics = 2
max_topics = 20
step_size = 1
topics_range = range(min_topics, max_topics, step_size)


# Set de validacion
num_of_docs = len(corpus)
corpus_sets = [gensim.utils.ClippedCorpus(corpus, int(num_of_docs*0.75)),
               corpus] # dividar para generar distintos corpus's

corpus_title = ['75% Corpus', '100% Corpus'] # nombre del corpus

model_results = {'Validation_Set': [],
                 'Topics': [],
                 'Coherence': []
                }
```

```
if 1 == 1:
    pbar = tqdm.tqdm(total=(len(topics_range)*len(corpus_title)))

    # iterar a traves de los copus's generados
    for i in range(len(corpus_sets)):
        # iterar el numero de topicos
        for k in topics_range:
            # obtener la medida de coherencia para los distintos modelos ajustasdos
            cv = compute_coherence_values(corpus=corpus_sets[i], dictionary=id2word,
                                          k=k)
            # gaurdar los resultados en formato CSV
            model_results['Validation_Set'].append(corpus_title[i])
            model_results['Topics'].append(k)
            model_results['Coherence'].append(cv)

            pbar.update(1)
    pd.DataFrame(model_results).to_csv('lda_tuning_results2.csv', index=False)
    pbar.close()
```

A.2. Código Naive Bayes.

```
# ============================================================================
# Librerias a ser Usadas.
# ============================================================================

import pandas as pd
import xml.etree.ElementTree as et
import glob
import os
import matplotlib.pyplot as plt
import re
import nltk
import seaborn as sns
import numpy as np
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from nltk.stem import SnowballStemmer
from nltk import word_tokenize
from sklearn import feature_extraction, model_selection, naive_bayes, pipeline, manifold, preprocessing,feature_sel


stemmer = SnowballStemmer('spanish')
lst_stopwords = nltk.corpus.stopwords.words("spanish")
spanish_stopwords = stopwords.words('spanish')
os.chdir(r"C:\Users\manuel.diaz\Documents\TASS")
```

```
#%%

# ========================================================================
# Importacion de datos y Creacion de DataFrame
# ========================================================================
df_cols = ["tweetid","user","content","date","lang","polarity"]
rows = []

# Funcion para la importacion de los datos
#en formato XML
def parse_XML(xml_file, df_cols):

    xtree = et.parse(xml_file)
    xroot = xtree.getroot()
    rows = []

    for node in xroot:
        tweetid = node.find("tweetid").text
        user = node.find("user").text
        content = node.find("content").text
        date = node.find("date").text
        lang= node.find("lang").text
        polarity = node.find("sentiment").find("polarity").find("value").text

        rows.append({"tweetid":tweetid, "user":user,"content":content,"date":date,
                     "lang":lang,"polarity":polarity})

    out_df = pd.DataFrame(rows, columns=df_cols)

    return out_df

files = glob.glob( "*.xml")

li = []
for filename in files:
    df = parse_XML(filename,df_cols)
    li.append(df)

data = pd.concat(li, axis=0, ignore_index=True)
data = data[data["polarity"] != 'NEU']


#remover links
tweets_corpus = data[-data.content.str.contains('^http.*$')]

#%%
```

```
# ================================================================================
# Crear una grafica de los unigramas mas comunes
# ================================================================================

cv = CountVectorizer(analyzer = 'word',
                lowercase = True,
                stop_words = spanish_stopwords)
words = cv.fit_transform(tweets_corpus.content)

sum_words = words.sum(axis=0)

words_freq = [(word, sum_words[0, i]) for word, i in cv.vocabulary_.items()]
words_freq = sorted(words_freq, key = lambda x: x[1], reverse = True)

frequency = pd.DataFrame(words_freq, columns=['word', 'freq'])

frequency.head(30).plot(x='word', y='freq', kind='bar', figsize=(15, 7), color = 'blue')
plt.title("Most Frequently Occuring Words - Top 30")
#%%

# ================================================================================
# funcion que permite generar una grafica para
# dos o tres n-gramas
# ================================================================================
from nltk import ngrams
def n_gram(text , n):
    n_grams = ngrams(text.split(), n)
    return n_grams

sentences = []
df_frequency = []

def create_plot(num):
    frequency(n_gram(num)) # Send "num" parameter to "n_gram func." and send the result to "frequency func."

    gram_frame = pd.DataFrame(sentences)       # gram_frame is the data frame to store grams and freq.

    gram_frame['frequencies'] = df_frequency
    if num == 2:
        gram_frame.columns = ['first', 'second', 'frequencies']
    if num == 3:
        gram_frame.columns = ['first', 'second', 'third', 'frequencies']

    gram_frame.sort_values("frequencies", axis=0, ascending=False, inplace=True, na_position='last')

    gram_frame = gram_frame.head(20)            # Only take the top 20 of gram_frame

    total = sum(df_frequency)
```

```
    gram_frame["ratio"] = gram_frame['frequencies'].div(total)   # Additional, ratio is added

    plt.rcdefaults()
    fig, ax = plt.subplots()

    if num == 2:
        grams = gram_frame["first"] + " " + gram_frame["second"]
    if num == 3:
        grams = gram_frame["first"] + " " + gram_frame["second"] + " " + gram_frame["third"]

    # Create plot
    y_pos = np.arange(len(grams))
    performance = gram_frame["frequencies"]

    ax.barh(y_pos, performance)
    ax.set_yticks(y_pos)
    ax.set_yticklabels(grams)
    ax.invert_yaxis()  # labels read top-to-bottom
    ax.set_xlabel('Frequency')
    ax.set_title('2-3 grams')

    plt.show()
    display(gram_frame)



#%%
# ================================================================================
# Funcion para una limpieza efectiva de los datos
# ================================================================================



def utils_preprocess_text(text, flg_stemm=False, lst_stopwords=None):

    text = re.sub(r'[^\w\s]', '', str(text).lower().strip())

    lst_text = word_tokenize(text)

    if lst_stopwords is not None:
        lst_text = [word for word in lst_text if word not in
                    lst_stopwords]

    if flg_stemm == True:
        ps = stemmer
        lst_text = [ps.stem(word) for word in lst_text]

    text = " ".join(lst_text)

    return text
```

```
data["clean text"] = data["content"].apply(lambda x:
                                          utils_preprocess_text(x,flg_stemm=True,
                                                                lst_stopwords=lst_stopwords))


#%%

# =============================================================================
# Grafica de las polaridades por recuento
# =============================================================================
fig, ax = plt.subplots()
fig.suptitle("polarity", fontsize=12)
data["polarity"].reset_index().groupby("polarity").count().sort_values(by=
       "index").plot(kind="barh", legend=False,
        ax=ax).grid(axis='x')
plt.show()

#%%

# =============================================================================
# division de los datos en datos de prueba y de testeo
# ademar de vectorizarlos y crear el corpus.
# =============================================================================
dtf_train, dtf_test = model_selection.train_test_split(data, test_size=0.1,
                                                       random_state=0) #semilla para resultados reproducibles

y_train = dtf_train["polarity"].values
y_test = dtf_test["polarity"].values


vectorizer = feature_extraction.text.CountVectorizer(max_features=10000,
                                                     ngram_range=(1,2))

corpus = dtf_train["clean text"]
vectorizer.fit(corpus) # se debe poner .strip() cuando se tokeniza
X_train = vectorizer.transform(corpus)
dic_vocabulary = vectorizer.vocabulary_
#%%

# =============================================================================
# tratar cada categoría como binaria (por ejemplo, la categoría "Tecnología" es 1 para las noticias
# tecnológicas y 0 para las demás);
# realizar una prueba de chi-cuadrado para determinar si una característica y el
# objetivo (binario) son independientes;
# mantenga solo las características con un cierto valor p de la prueba de chi-cuadrado.
# =============================================================================

y = dtf_train["polarity"]
```

```
X_names = vectorizer.get_feature_names()
p_value_limit = 0.95


dtf_features = pd.DataFrame()
for cat in np.unique(y):
    chi2, p = feature_selection.chi2(X_train, y==cat)
    dtf_features = dtf_features.append(pd.DataFrame(
                   {"feature":X_names, "score":1-p, "y":cat}))
    dtf_features = dtf_features.sort_values(["y","score"],
                    ascending=[True,False])
    dtf_features = dtf_features[dtf_features["score"]>p_value_limit]
X_names = dtf_features["feature"].unique().tolist()

vectorizer = feature_extraction.text.TfidfVectorizer(vocabulary=X_names)
vectorizer.fit(corpus)
X_train = vectorizer.transform(corpus)
dic_vocabulary = vectorizer.vocabulary_
#%%

# ============================================================================
# Ajuste del modelo Bayes Ingenuo.
# ============================================================================

classifier = naive_bayes.MultinomialNB()


model = pipeline.Pipeline([("vectorizer", vectorizer),
                           ("classifier", classifier)])

model["classifier"].fit(X_train, y_train)


X_test = dtf_test["clean text"].values
predicted = model.predict(X_test)
predicted_prob = model.predict_proba(X_test)
predicted = model.predict(X_test)

dtf_test["predictions"] = predicted

dtf_test[["content","predictions"]]

pd.DataFrame([X_test, predicted])


#%%
# ============================================================================
# Evaluacion del Modelo de Bayes Ingenuo.
# ============================================================================
```

```
# ____________Evaluar Performace.

classes = np.unique(y_test)
y_test_array = pd.get_dummies(y_test, drop_first=False).values


## Accuracy, Precision, Recall

accuracy = metrics.accuracy_score(y_test, predicted)
auc = metrics.roc_auc_score(y_test, predicted_prob[:, 1])


print("Accuracy:",  round(accuracy,2))
print("Auc:", round(auc,2))
print("Detail:")
print(metrics.classification_report(y_test, predicted))


## Graficar Matriz de Confusion.

cm = metrics.confusion_matrix(y_test, predicted)
fig, ax = plt.subplots()
sns.heatmap(cm, annot=True, fmt='d', ax=ax, cmap=plt.cm.Blues,
            cbar=False)
ax.set(xlabel="Pred", ylabel="True", xticklabels=classes,
       yticklabels=classes, title="Confusion matrix")
plt.yticks(rotation=0)



## Graficar Curva ROC y Curvas de Precision-Recall

fig, ax = plt.subplots(nrows=1, ncols=2)
for i in range(len(classes)):
    fpr, tpr, thresholds = metrics.roc_curve(y_test_array[:,i],
                           predicted_prob[:,i])
    ax[0].plot(fpr, tpr, lw=3,
              label='{0} (area={1:0.2f})'.format(classes[i],
                              metrics.auc(fpr, tpr))
               )
ax[0].plot([0,1], [0,1], color='navy', lw=3, linestyle='--')
ax[0].set(xlim=[-0.05,1.0], ylim=[0.0,1.05],
          xlabel='False Positive Rate',
          ylabel="True Positive Rate (Recall)",
          title="Receiver operating characteristic")
ax[0].legend(loc="lower right")
ax[0].grid(True)


for i in range(len(classes)):
```

```
    precision, recall, thresholds = metrics.precision_recall_curve(
                 y_test_array[:,i], predicted_prob[:,i])
    ax[1].plot(recall, precision, lw=3,
               label='{0} (area={1:0.2f})'.format(classes[i],
                                   metrics.auc(recall, precision))
              )
ax[1].set(xlim=[0.0,1.05], ylim=[0.0,1.05], xlabel='Recall',
          ylabel="Precision", title="Precision-Recall curve")
ax[1].legend(loc="best")
ax[1].grid(True)

plt.tight_layout()
plt.show()
```

A.3. Streamming de Datos Twitter API.

```
# ========================================================================
# librerias necesarias.
# ========================================================================

import tweepy
import json
import csv
import os
os.chdir(r"C:\Users\manuel.diaz\Documents\T")

name = "Bogota"


# ========================================================================
# Key y Token ID's
# ========================================================================

consumer_key = "#################"
consumer_secret = "#################"
acces_token = "############################"
acces_token_secret = "######################"


# ========================================================================
#  Conexion a la API de Twitter.
# ========================================================================

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(acces_token, acces_token_secret)

api = tweepy.API(auth_handler=auth,wait_on_rate_limit=True,
```

```
                wait_on_rate_limit_notify=True)

# ================================================================================
#  Crear Funcion para el Streaming de los tweets.
# ================================================================================

# crear una clase
class CustomStreamListener(tweepy.StreamListener):

    # status del streaming
    def on_status(self, status):
        try: # texto de mas de 140 caracteres
            text = status.extended_tweet["full_text"]
        except AttributeError:
            text = status.text # texto de menos de 140 caracteres
        print (status.author.screen_name, status.created_at, text)
        # Guardando los datos
        with open("OutputStreaming_{}.csv".format(name), 'a',encoding="utf-8") as f:
            writer = csv.writer(f)
            writer.writerow([status.author.screen_name, status.created_at, text,status.entities["hashtags"],
                             status.coordinates])

    # Imprimir error en alguna parte de las credenciales
    def on_error(self, status_code):
        print >> sys.stderr, 'Encountered error with status code:', status_code
        return True
    # Imprimir error si toma demasiado tiempo la conexion
    def on_timeout(self):
        print >> sys.stderr, 'Timeout...'
        return True # Don't kill the stream
    # imprimir si existe algun error de alguna otra excepcion
    def on_exception(self, exception):
        print(exception)
        return

# Definir donde y como se guarda los resultados del streaming de tweets
import io
with io.open("OutputStreaming_{}.csv".format(name), 'w', encoding="utf-8") as f:
    writer = csv.writer(f)
    writer.writerow(['Author', 'Date', 'Text',"hashtags","coordinates"])

# comenzar el streaming
streamingAPI = tweepy.streaming.Stream(auth, CustomStreamListener())
# Definir el filtro por ubicacion.
streamingAPI.filter(locations=[-74.89066632,3.72434499,-73.05083455,5.83670244])
```