GOBIERNO DE MÉXICO | CONAHCYT CONSEJO NACIONAL DE HUMANIDADES CIENCIAS Y TECNOLOGÍAS | INFOTEC

BIBLIOTECA INFOTEC
VISTO BUENO DE TRABAJO TERMINAL

Maestría en Ciencia de Datos e Información
(MCDI)

Ciudad de México, a 1 de julio de 2024

UNIDAD DE POSGRADOS
PRESENTE

Por medio de la presente se hace constar que el trabajo de titulación:

"Análisis y optimización de tópicos con enfoque en el Sistema Bancario Mexicano utilizando BERTOPIC"

Desarrollado por el alumno: **Juan Antonio Torres Veayra**, bajo la asesoría del **Dr. José Ortiz Bejar** cumple con el formato de Biblioteca, así mismo, se ha verificado la correcta citación para la prevención del plagio; por lo cual, se expide la presente autorización para entrega en digital del proyecto terminal al que se ha hecho mención. Se hace constar que el alumno no adeuda materiales de la biblioteca de INFOTEC.

No omito mencionar, que se deberá anexar la presente autorización al inicio de la versión digital del trabajo referido, con el fin de amparar la misma.

Sin más por el momento, aprovecho la ocasión para enviar un cordial saludo.

Mtro. Carlos Josué Lavandeira Portillo
Director Adjunto de Innovación y Conocimiento

Jah

CJLP/jah

C.c.p. Felipe Alfonso Delgado Castillo.- Gerente de Capital Humano.- Para su conocimiento.
Juan Antonio Torres Veayra.- Alumno de la Maestría en Ciencia de Datos e Información.- Para su conocimiento.

INFOTEC CENTRO DE INVESTIGACIÓN E INNOVACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

“ANÁLISIS Y OPTIMIZACIÓN DE TÓPICOS CON ENFOQUE EN EL SISTEMA BANCARIO MEXICANO UTILIZANDO BERTOPIC”

PROYECTO APLICATIVO
Que para obtener el grado de
MAESTRO EN CIENCIA DE DATOS E INFORMACIÓN

Presenta:

Juan Antonio Torres Veayra

Asesor:

Dr. José Ortiz Béjar

Ciudad de México, marzo de 2024

MCDI
Maestría en Ciencia de Datos e Información

INFOTEC
POSGRADOS

Tabla de contenido

Índice de figuras

Índice de cuadros

Índice de cuadros

Siglas y abreviaturas

API	Interfaz de programación de aplicaciones, por sus siglas en inglés, Application Programming Interface.
BERT	Representación de codificador bidireccional de transformadores, por sus siglas en inglés Bidirectional Encoder Representations from Transformers.
CRISP-DM	Por sus siglas en inglés Cross Industry Standard Process for Data Mining.
c-TF-IDF	Frecuencia de término frecuencia inversa de documento basado en clases, por sus siglas en inglés Class-based Term Frequency – Inverse Document Frequency.
CV	Vector de coherencia, por sus siglas en inglés Coherence Vector.
fANOVA	Análisis de varianza funcional, del inglés, Functional ANalysis of VAriance.
GPT	Por sus siglas en inglés Generative Pre-Trained Transformer.
HDBSCAN	Agrupación espacial jerárquica basada en la densidad de aplicaciones con ruido, por sus siglas en inglés Hierarchical Density-Based Spatial Clustering of Applications with Noise.
LDA	Asignación Latente de Dirichlet, por sus siglas en inglés Latent Dirichlet Allocation.
NFKC	Por sus siglas en inglés Normalization Form KC.
NFM	Factorización no negativa de matrices, por sus siglas en inglés Non-negative Matrix Factorization.
NPMI	Información mutua normalizada por puntos, por sus siglas en inglés Normalized Pointwise Mutual Information.
NPS	Índice de promotores neto, por sus siglas en inglés Net Promoter Score.
NTM	Modelos de temas neuronales, por sus siglas en inglés Neural Topic Model.
OCTIS	Por sus siglas en inglés Optimizing and Comparing Topic models

	Is Simple.
PCA	Análisis de componentes principales, por sus siglas en inglés Principal Component Analysis.
PMI	Información mutua puntual, por sus siglas en inglés Pointwise Mutual information.
SBERT	Representación de codificador bidireccional de transformadores de oraciones, por sus siglas en inglés Sentence Bidirectional Encoder Representations from Transformers.
SEMMA	Muestreo, exploración, modificación, modelado, evaluación, por sus siglas en inglés Sample, Explore, Modify, Model, Assess.
SVD	Descomposición de valores singulares, por sus siglas en inglés Singular Value Descomposition.
TF-IDF	Frecuencia de término – frecuencia inversa de documento, por sus siglas en inglés Term Frequency – Inverse Document Frequency.
UMAP	Aproximación y proyección de colector uniforme, por sus siglas en inglés Uniform Manifold Approximation and Projection.
URL	Localizador de Recursos Uniforme, por sus siglas en inglés Uniform Resource Locator.

Glosario

Agglomerative Clustering. Algoritmo de agrupamiento de tipo jerárquico que utiliza un enfoque de agrupamiento ascendente, empieza con grupos de una observación y los fusiona sucesivamente.

Algoritmo de agrupamiento. Algoritmo de aprendizaje automático no supervisado que organiza y clasifica observaciones en grupos basados en similitud o algún patrón en común.

Aprendizaje automático. Rama de la inteligencia artificial y ciencias de la computación que se enfoca en usar datos y algoritmos estadísticos, para generar algoritmos capaces de imitar la forma de aprendizaje humano, aprender de los datos, generalizar y realizar tareas sin una programación explícita.

Bidirectional Encoder Representations from Transformers (BERT). Modelo de aprendizaje automático para procesamiento de lenguaje natural, desarrollado por Google AI en 2018.

BERTopic. Técnica de modelo de tópicos que aprovecha el uso de transformadores, de tipo BERT u otros, y uso de Class-based Term Frequency – Inverse Document Frequency para el ponderado de pesos.

Coherencia. Medida para evaluar la calidad de los tópicos obtenidos de un modelo de tópicos. Un valor alto indica que las palabras que conforman un tópico son semánticamente similares entre ellas, haciendo que el tópico sea significativo e interpretable.

Corpus. Conjunto de documentos de texto que se utiliza como entrada para un modelo de procesamiento de lenguaje natural.

CountVectorizer. Algoritmo utilizado para convertir una colección de documentos en una representación numérica matricial llamada matriz documento-término donde

las filas representan documentos, las columnas representan token o palabras y valores números de cada celda representa la frecuencia

Class-based Term Frequency–Inverse Document Frequency (c-TF-IDF). Versión ajustada del algoritmo Term Frequency–Inverse Document Frequency (TF-IDF) que funciona a nivel clase.

Diversidad. Medida para evaluar la calidad de los tópicos en un modelo de tópicos, indica el grado con el que un tópico engloba características únicas de sí mismo y que no están presentes en otros tópicos, un tópico con diversidad alta tendrá palabras únicas que no aparecen frecuentemente en otros tópicos.

Embedding. Representación de textos o documentos como vectores en un espacio de alta dimensión.

Generative Pre-Trained Transformer (GPT). Tipo de modelo de aprendizaje automático de procesamiento de lenguaje natural, desarrollado por Open AI.

Hashtag. Palabra o frase precedida de un signo de almohadilla (#) utilizada en sitios y aplicaciones de redes sociales, para identificar contenidos digitales sobre un tema en específico.

Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN). Algoritmo de agrupamiento jerárquico utilizado en el aprendizaje no supervisado. Es una extensión del algoritmo (Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

Hiperparámetro. Valor o ajuste de configuración que se determina antes de entrenar un modelo de aprendizaje automático.

K-Means. Algoritmo de agrupamiento para aprendizaje no supervisado.

Latent Dirichlet Allocation (LDA). Modelo estadístico utilizado en el procesamiento natural del lenguaje y aprendizaje automático para el modelado de tópicos.

Mención. Dentro de la red social X, parte del mensaje que hace referencia a otro usuario, precedido del símbolo de arroba (@).

Métricas de evaluación. Son medidas cuantitativas para evaluar el desempeño y eficacia de un modelo estadístico o de aprendizaje automático, proporcionan información sobre el modelo y ayudan a comparar diferentes modelos.

Minería de texto, o minería de datos de texto. Es el proceso de transformar texto no estructurado en un formato estructurado para identificar patrones significativos y nueva información (IBM Corporation, 2024).

Modelado de tópicos. Técnica del aprendizaje automático no supervisado que consiste en encontrar patrones semánticos en documentos de texto e identificar de manera automática los tópicos que contienen.

Non-negative Matrix Factorization (NFM). Método estadístico utilizado en modelado de tópicos para realizar reducción dimensional del corpus de entrada. Utiliza análisis factorial para dar menor peso a las palabras que tienen menos coherencia.

Normalized Pointwise Mutual Information (NPMI). Es la versión normalizada de la medida de asociación información mutua puntual. Se define en el intervalo [-1,1], donde un valor de -1 son eventos que nunca ocurren juntos, un valor de 0 son eventos independientes y +1 son eventos que siempre coocurren.

Neural Topic Models (NTM). Conjunto de modelos de redes neuronales utilizados para aplicaciones de minería de texto como lo son análisis de texto, generación de texto, recomendación de contenido y modelado de tópicos.

Optimización bayesiana. Técnica utilizada en aprendizaje automático que hace uso de modelos probabilísticos para encontrar de manera eficiente los hiperparámetros de un modelo.

Optuna. Herramienta y marco de trabajo de optimización de hiperparámetros de código abierto, diseñado para automatizar el proceso de ajuste de hiperparámetros de modelos de aprendizaje automático.

Palabras vacías (stop words). Dentro del contexto de procesamiento de lenguaje natural, palabras que son filtradas desde un inicio por no proporcionar mucha información, comúnmente son artículos, preposiciones y pronombres.

Principal Component Analysis (PCA). Método estadístico para reducción dimensional, que a la vez retiene la mayor información posible.

Pipeline. En Python, en una secuencia de pasos de procesamiento de datos, donde cada paso recibe los datos de un paso anterior, realiza alguna operación o transformación y paso el resultado al siguiente paso.

Pointwise Mutual information (PMI). Es una medida de asociación usada en estadística, teoría de la probabilidad y teoría de la información. Compara la probabilidad de que dos eventos ocurran simultáneamente.

Python. Lenguaje de programación de alto nivel, interpretado, orientado a objetos y de propósito general que se enfoca en la legibilidad del código (Python Software Foundation, 2024).

SCiPY. Librería de Python de código abierto que proporciona una amplia gama de capacidades de computación científica en Python.

Scikit-learn. Librería de Python de código abierto utilizada para el aprendizaje automático, que simplifica el proceso de implementar modelos de aprendizaje automático y estadísticos en Python.

Term Frequency – Inverse Document Frequency (TF-IDF). Es una medida estadística utilizada en el procesamiento del lenguaje natural que mide la importancia de un término dentro de un documento en relación con una colección de documentos.

Tweet. Publicación en la red social X (anteriormente Twitter).

Uniform Manifold Approximation and Projection (UMAP). Técnica de reducción dimensional no lineal que tiene como objetivo capturar la estructura global y local de los datos.

Unicode. Norma internacional de codificación para diferentes idiomas y alfabetos, por la que se asigna a cada letra, dígito o símbolo un valor numérico único que sirve para diferentes plataformas y lenguajes.

Introducción

La medición de la satisfacción de los clientes, y más en específico la opinión que tienen sobre el servicio, es de enorme importancia dentro del sector bancario, ya que al ser mayormente un sector proveedor de servicios, uno de los diferenciadores clave es la calidad en la atención a clientes, ya sea de manera presencial en una sucursal, por teléfono en un centro de atención a clientes, mediante una aplicación móvil o cualquier otro medio de interacción entre el usuario y el banco.

Existen una gran variedad de técnicas y metodologías para llevar a cabo esta medición, la mayor parte de ellas basadas en encuestar al usuario ya sea de manera presencial, telefónica o mediante medios electrónicos, y aunque estas encuestas son una excelente herramienta, también pueden presentar uno o varios inconvenientes. Realizar encuestas de la forma tradicional conlleva ciertos desafíos e inconvenientes, tales como:

- Costo. Llevar a cabo una encuesta a nivel nacional involucra entrevistar de cientos a miles de personas, aun y cuando se utilicen encuestas telefónicas o en medios electrónicos esto por lo general es costoso. La práctica común es contratar a una agencia encuestadora y lo cual queda fuera del alcance para instituciones más pequeñas con presupuestos limitados.

- Demora en la obtención de resultados. Derivado del costo, las encuestas por lo general se realizan de modo trimestral o mensual, lo que demora la obtención de resultados. En el dinamismo empresarial actual, tener resultados en tiempo real es una ventaja competitiva.

- Granularidad de los resultados. Para instituciones con presencia nacional, es sumamente importante obtener resultados regionales y estratificados, ya que de este modo se pueden crear estrategias focalizadas para grupo de clientes con características similares. En la práctica, implementar esto implica aplicar

muchas encuestas para conservar la representatividad estadística en niveles de granularidad más pequeños, lo cual no siempre es posible.

Dado que el uso de análisis de sentimientos y opiniones en redes sociales no se ve afectado por los inconvenientes previamente mencionados: el costo es relativamente bajo, la información se puede obtener en tiempo real y la cantidad de comentarios en redes sociales es generalmente mayor a lo que se puede obtener realizando encuestas; resulta conveniente utilizar estos datos como complemento a las mediciones de satisfacción actuales; esto cuando sea posible o como un reemplazo cuando existan limitantes financieras.

El primer objetivo de este trabajo es evaluar la factibilidad de utilizar comentarios en redes sociales, en específico de la red social X (anteriormente Twitter), como medio alterno para evaluar la opinión de los clientes, mediante el uso de la técnica de modelado de tópicos.

Hoy en día existen una gran variedad de modelos para la obtención de tópicos, desde los clásicos Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan, 2003) y Non-negative Matrix Factorization (NFM) (Yan, Guo, Liu, Cheng, & Wang, 2013) hasta modelos de reciente creación como lo son Bidirectional Encoder Representations from Transformers (BERT) (Devlin, Chang, Lee, & Toutanova, 2019) y Generative Pre-Trained Transformer (GPT) (Radford, Narasimhan, Salimans, & Sutskever, 2018) los cuales son posibles dado el incremento de la capacidad computacional en los últimos años, que ha permitido entrenar modelos con conjuntos de datos cada vez más grandes, en el rango de billones a trillones de parámetros de entrada.

Con esta nueva generación de algoritmos y técnicas para el modelado de datos surge BERTopic, el cual es un marco de trabajo de modelado de tópicos modular que hace uso de estos nuevos algoritmos y herramientas y permite la fácil integración de modelos transformadores pre-entrenados, como lo son BERT o GPT, para la generación de tópicos coherentes y fácilmente interpretables.

Un segundo objetivo de este trabajo es analizar la viabilidad del uso de BERTopic para el modelado de tópicos en textos cortos del sector bancario, obtenidos de la red social Twitter, y a su vez explorar las opciones y técnicas de optimización de

hiperparámetros que permitan mejorar su rendimiento y la calidad de los tópicos generados.

Al ser BERTopic un modelo modular que integra un grupo de algoritmos cada uno con su propio conjunto de parámetros de entrada y configuración, resulta complejo determinar los valores óptimos estos, comprender las interrelaciones e implicaciones de modificar un hiperparámetro y cómo esto afecta a los pasos subsecuentes. La complejidad es el resultado de las miles o cientos de miles de configuraciones posibles. Es por este motivo, que en este trabajo se propone la utilización de técnicas de optimización bayesiana en 2 etapas, primero para identificar los algoritmos adecuados y posteriormente para optimizar los hiperparámetros de cada algoritmo.

Como parte del trabajo, también se valorarán las opciones actuales de métricas de desempeño, coherencia y diversidad, para la evaluación de modelos de tópicos, con el objetivo de verificar que tan efectivas son estas para guiar un proceso de optimización automático en la generación de tópicos de calidad y utilidad práctica.

Capítulo 1

Marco teórico y antecedentes

Capítulo 1. Marco teórico y antecedentes

1.1 Medición de opinión de clientes

Hoy en día existen numerosas metodologías para la medición de la opinión y la satisfacción de los clientes respecto a un producto o servicio. Estas metodologías dependen de lo que se entiende por “satisfacción del cliente” y sobre todo en el objetivo de la medición. Estas mediciones dependen en gran medida de factores tales como el tipo de aplicación de la encuesta (por ejemplo, presencial o electrónica), el tamaño y representatividad estadística de la muestra de clientes encuestados, el momento en el ciclo de vida del cliente en el que se obtiene la medición (no es lo mismo una medición realizada después de una interacción a una realizada aleatoriamente), el tamaño y número de preguntas de la encuesta, entre otros factores muy diversos que otorgan ciertas ventajas y desventajas.

Algunas de las alternativas actuales más ampliamente utilizadas son:

1.1.1 Modelo SERVQUAL

Modelo propuesto por Parasuraman, Zeithaml, y Berry en 1985 cuyo objetivo es medir la satisfacción del cliente respecto a una empresa, es un modelo ampliamente usado por instituciones bancarias y en general por empresas de servicio. Consiste de un cuestionario con una serie de preguntas donde se solicita al cliente evaluar un producto o servicio en una escala de 1 a 7.

Parasuramen, Zeithaml y Berry (1988) describen el modelo SERVQUAL en base a las siguientes 5 dimensiones para medir la satisfacción del cliente:

1. Tangibles: aspectos tangibles y físicos tales como tiendas, instalaciones, equipo físico y apariencia del personal (Parasuramen, Zeithaml, & Berry, 1988).
2. Fiabilidad: habilidad de cumplir con el servicio ofrecido (Parasuramen, Zeithaml, & Berry, 1988).
3. Capacidad de respuesta: voluntad de ayudar al cliente y ayudar rápidamente (Parasuramen, Zeithaml, & Berry, 1988).

4. Garantía: Conocimientos y cortesía de los empleados, así como su habilidad de ofrecer confianza (Parasuramen, Zeithaml, & Berry, 1988).
5. Empatía: mostrar empatía y atención individual a cada uno de los clientes (Parasuramen, Zeithaml, & Berry, 1988).

1.1.2 Network Promoter Score

Métrica desarrollada por Fred Reichheld en 2003 como una alternativa para medir la satisfacción y lealtad. De acuerdo con Reichheld (2015), para una empresa la lealtad de sus clientes es el principal factor que genera crecimiento, y por consiguiente el mejor camino hacia el crecimiento y rentabilidad consiste en obtener clientes leales.

El planteamiento de Network Promoter Score (NPS), se basa en el principio “Keep it Simple”, es decir, realizar la medición de la manera más simple posible; es por eso que la encuesta NPS consta de una sola pregunta: “En una escala de 0 a 10, ¿Con qué probabilidad recomendaría nuestra empresa a un amigo?”.

Como primer paso de la construcción de la métrica NPS se generan 3 grupos clientes de acuerdo a la evaluación otorgada: promotores (con evaluaciones 9 o 10), pasivos (7 o 8) y detractores (6 o menos). Posteriormente se resta el porcentaje de detractores al porcentaje de promotores, con esto se obtiene un resultado número en el intervalo [-100,100].

Si bien no existen límites específicos respecto a que es un resultado positivo o negativo, esta métrica resulta de bastante utilidad al compararla con competidores dentro de la misma industria o con la evolución de la misma respecto al tiempo.

1.1.3 Normas de la familia ISO 9000

Son un conjunto de normas generadas por la Organización Internacional de Normalización (International Organisation for Standardisation, 2015) para el correcto control y gestión de la calidad. Aunque su alcance es mucho más amplio que simplemente la medición de la satisfacción tiene como uno de sus elementos más importantes el conocer al cliente y establecer los procesos o canales para la medición constante y continua de la satisfacción. Estas normas resaltan la

importancia dentro de una organización de realizar mediciones de satisfacción como parte de los pilares para la toma de decisiones.

1.2 Metodologías de Minería de Datos

Como lo mencionan Azevedo y Santos (2008) el crecimiento y consolidación de los últimos años en el área de minería de datos se ha visto reflejado en los esfuerzos tanto académicos como de la industria privada para establecer estándares y marcos de trabajo que sirvan como referencia en el desarrollo de proyectos de Minería de Datos y Aprendizaje Automático.

De entre las metodologías que destacan por su aceptación y uso extenso se encuentran:

1.2.1 Cross Industry Standard Process for Data Mining (CRISP-DM)

Originalmente desarrollado por un consorcio de empresas compuesto por DaimlerChrysler AG, SPSS, NCR Corporation y OHRA (Wirth & Hipp, 2000). Como se aprecia en la Figura *1*, la metodología CRISP-DM contempla la división en 6 fases del ciclo de vida de un proyecto de minería de datos:

1. Entendimiento de negocio. Esta primera fase se enfoca en el entendimiento de los objetivos del proyecto y los requerimientos desde una perspectiva de negocio.
2. Entendimiento de los datos. Implica recopilar los primeros datos y realizar la exploración de los mismos. Buscar tendencias y patrones de interés; y crear hipótesis que se deseen comprobar con los datos.
3. Preparación de los datos. Construcción del conjunto de datos para el modelado. Incluye las tareas relacionadas a la limpieza de datos, selección de atributos, transformaciones y construcción de nuevos atributos.
4. Modelado. Uso de herramientas analíticas y técnicas de modelado de datos, calibración de los parámetros de estos modelos para obtener resultados óptimos.

5. Evaluación. Fase de evaluación de los resultados obtenidos del modelo de datos. Se incluye la revisión de la calidad de los resultados obtenidos y la selección del modelo más adecuado que se ajuste a los objetivos del proyecto.
6. Despliegue. Dependiendo del proyecto puede implicar desde generar un reporte con los resultados hasta desplegar un modelo de Aprendizaje Automático para uso en tiempo real. El objetivo de este paso es generar valor de los datos y el proyecto de minería de datos.

Aunque cada paso es dependiente al paso previo, el modelo es flexible y permite en cualquier momento regresar a pasos anteriores. CRISP-DM se basa en la mejora continua, por lo cual fundamenta que un proyecto de minería de datos no necesariamente concluye una vez que la solución es desplegada, sino que recomienda iterar varias veces el ciclo completo.

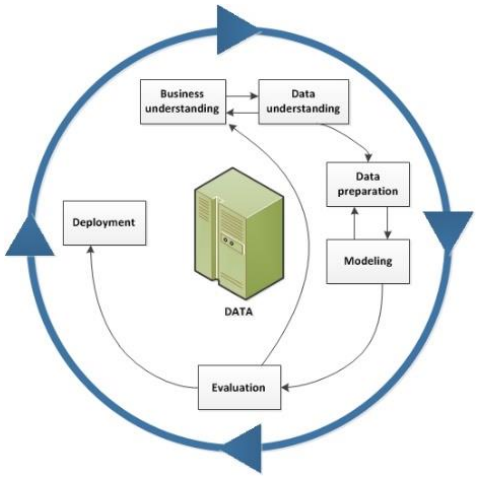


Figura 1. Modelo CRISP-DM.

Fuente: *(IBM Corporation, 2024)*

1.2.2 Sample, Explore, Modify, Model, Assess (SEMMA).

Desarrollado por (SAS Institute Inc., 2024) originalmente para su propio software Enterprise Miner, ha sido adoptado por múltiples industrias y negocios. Tiene la ventaja de ser sencillo de entender y aplicar.

Su nombre viene de las iniciales en inglés de las 5 etapas que lo conforma, las cuales son:

1. Muestreo, de los datos, para facilitar su posterior uso. Para modelos de aprendizaje supervisado incluye crear particiones de datos para entrenamiento y validación.
2. Exploración, en este paso el objetivo es buscar tendencias, anomalías y relaciones de interés. Incluye el adquirir conocimiento de los datos.
3. Modificación, consiste en transformar los datos, seleccionar atributos de interés y/o crear nuevos atributos a partir de los ya existentes.
4. Modelado, de los datos mediante herramientas analíticas, entre las que se encuentran los modelos de aprendizaje automático.
5. Evaluación, de los resultados obtenidos, el uso y calidad de los mismos.

Como se observa en la Figura *2*, la metodología es un proceso lineal de pasos consecutivos bien establecidos; sin embargo, al igual que la metodología CRISP-DM, esta se plantea como un proceso iterativo que se puede y sugiere repetir tantas veces como sea necesario para encontrar resultados satisfactorios.

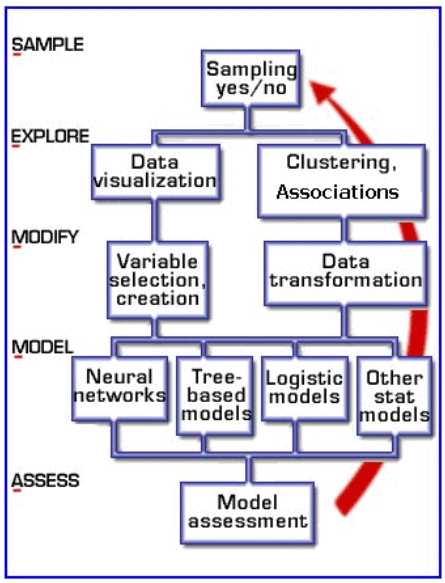


Figura 2. Modelo SEMMA

Fuente: *(SAS Institute Inc., 2024)*

1.3 Modelado de Tópicos

Dentro de la gran variedad de algoritmos y técnicas de construcción de modelos de aprendizaje automático, existe el campo del análisis de texto que se encarga de

estudiar, analizar y crear modelos de texto. También se le conoce como minería de texto y en general se encarga de buscar patrones en estructuras de texto usando la relación entre elementos del lenguaje como son: letras, palabras y/o documentos y el entendimiento del lenguaje humano a través de la semántica y las estructuras gramaticales propias de cada idioma.

Uno de los principales casos de uso de la minería de texto es el modelado de tópicos. En términos sencillos, el modelado de tópicos consiste en las metodologías y técnicas para llevar a cabo el descubrimiento de tópicos comunes dentro de una colección de documentos que por lo general es muy grande, también conocido como corpus.

Como lo explican Churchill y Singh (2022) esto se logra generando un resumen muy corto que recoge los temas más frecuentes presentes en el corpus, a este resumen conformado por las palabras más representativas de los documentos del corpus se le llama tópico. Un modelo de tópicos se puede definir como un modelo que toma como entrada una conjunto de documentos D, y genera como resultado un conjunto de tópicos T, que representan el contenido de D de manera precisa y coherente.

El modelado de tópicos es su variante más sencilla se puede describir como un modelo de aprendizaje automático no supervisado de agrupamiento, sin embargo, con el uso cada vez más extenso de las redes neuronales profundas y el incremento del poder de poder de cómputo nuevos modelos han surgido, los modelos conocidos como Neural Topic Model (NTM) (Zhao, et al., 2021).

De los modelos clásicos se puede destacar Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan, 2003), el cual es un modelo probabilístico donde cada documento es representado como una bolsa de palabras. Tiene la desventaja de no considerar el orden de las palabras por lo que descarta el contexto y la semántica del texto. Por otra parte, es fácil de implementar y utilizar, funcionando mejor con textos largos.

Otra técnica ampliamente utilizada es Non-negative Matrix Factorization (NFM) (Yan, Guo, Liu, Cheng, & Wang, 2013). Se ha encontrado que ofrece buenos resultados en textos cortos (Yan, Guo, Liu, Cheng, & Wang, 2013) (Shi, Kang, Choo, & Reddy, 2018), como lo son Tweets y comentarios en redes sociales.

En cuanto a los modelos NTM destaca BERTopic (Grootendorst M. P., 2022) por ser un modelo modular que usa técnicas de agrupamiento optimizadas y una variación del modelo Term Frequency – Inverse Document Frequency (TF-IDF) (Grootendorst M. , 2020). Se comporta mejor que otros modelos con textos cortos ya que es capaz de inferir el contexto.

Una ventaja adicional de BERTopic es que al ser modular permite la inclusión de nuevos modelos y técnicas cada vez más avanzadas conforme se desarrollan, es decir, es posible integrarlos al flujo de BERTopic de manera sencilla. La versatilidad de BERTopic permite desarrollar e incluir nuevos modelos específicos para casos de uso en concreto.

1.4 BERTopic

Modelo de tópicos desarrollado por Maarten Grootendorst (Grootendorst M. P., 2022). Se tomó la determinación de utilizar BERTopic para este proyecto por la flexibilidad que ofrece, la amplia variedad de parámetros y modelos disponibles para configurarlo que junto con el uso de la metodología propuesta por BERTopic permite la fácil integración de diferentes modelos de lenguaje pre-entrenados.

BERTopic está conformado por 6 módulos o pasos Figura *3*, cada uno de ellos intercambiables y configurables.

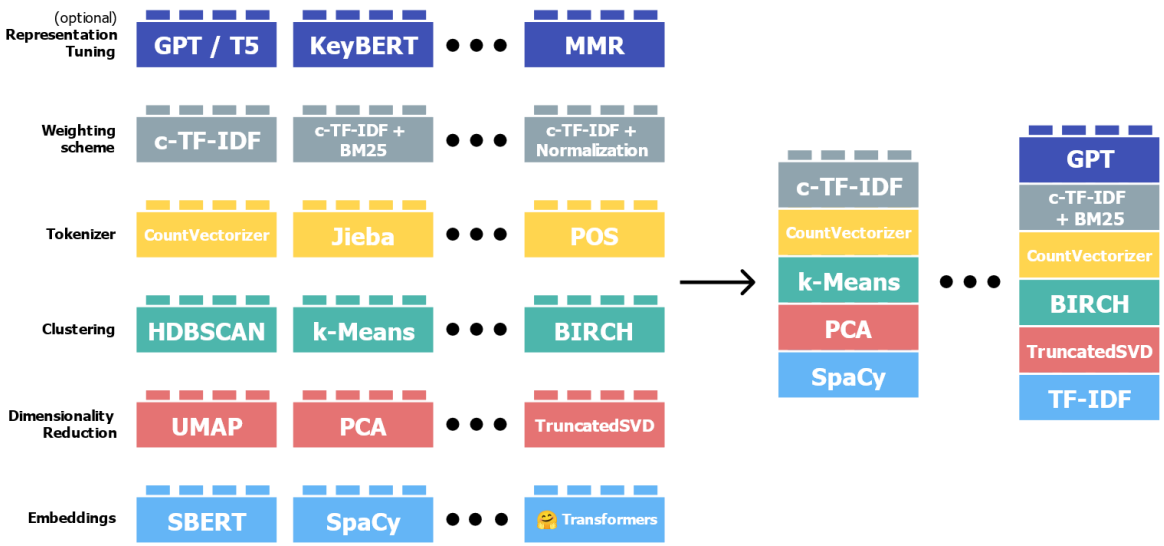


Figura 3. Módulos y construcción de BERTopic.

Fuente: (Grootendorst M. P., BERTopic, 2024)

1.4.1 Embedding

Es el proceso de generar una representación numérica a partir de los documentos de entrada. En modelos basados en bolsa de palabras como Latent Dirichlet Allocation (LDA) o Non-negative Matrix Factorization (NMF), se construye una matriz de alta dimensionalidad donde cada palabra única se representa como una columna y cada documento como una fila.

Los modelos más recientes conocidos como transformadores de oraciones son capaces de capturar la similitud semántica entre documentos y el contexto de las palabras, estos modelos son pre-entrenados con fuentes de datos muy grandes, los modelos más recientes ya cuentan con millones de parámetros. Entre ellos destaca Bidirectional Encoder Representations from Transformers (BERT) (Devlin, Chang, Lee, & Toutanova, 2019) desarrollado Google en 2018 y Generative Pre-Trained Transformer (GPT) (Radford, Narasimhan, Salimans, & Sutskever, 2018) cuya primera versión fue desarrollada por Open AI en 2018. Estos modelos se encuentran en constante evolución y mejora, por lo que es probable que sigan desarrollándose nuevos modelos cada vez más complejos, con más parámetros y con capacidades de entendimiento de contexto y similitud mayores.

En BERTopic se utiliza por defecto el modelo pre-entrenado de transformación de oraciones "all-MiniLM-L6-v2" (Reimers & Gurevych, 2019), sin embargo, se puede intercambiar por cualquier modelo de bolsa se palabras o transformador de oraciones.

1.4.2 Reducción dimensional

Por regla general, el paso de embedding genera una matriz de alta dimensionalidad, la cual es difícil de manipular, además de que el uso de datos dispersos en alta dimensionalidad, comúnmente no genera buenos resultados. Para eliminar estos obstáculos se utilizan algoritmos de reducción dimensional los cuales se encargan de transformar los datos para proyectarlos en una dimensión menor, lo cual, además de reducir el coste computacional; idealmente mejoraría los resultados a la vez que no se pierde la información contenida en los datos.

En BERTopic el algoritmo que se utiliza por defecto es Uniform Manifold Approximation and Projection (UMAP) (McInnes, Healy, & Melville, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, 2020). Sin embargo, se pueden utilizar otras alternativas como Principal Component Analysis (PCA) (Pearson, 1901), Truncated Singular Value Descomposition (Truncated SVD) (Hansen, 1987) o cualquiera de los algoritmos de reducción dimensional implementados en las librerías scikit-learn (Pedregosa, et al., 2011) o SCiPY (Virtanen, et al., 2020).

1.4.3 Agrupamiento

Este es el paso central del modelado de tópicos, ya que en esta etapa es donde se agrupan documentos similares, los cuales son representados como embeddings para formar los tópicos. En BERTopic el algoritmo por defecto es Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) (McInnes, Healy, & Astels, hdbscan: Hierarchical density based clustering, 2017), sin embargo, existen otras alternativas como K-Means (Lloyd, 1982) (MacQueen, 1967) o Agglomerative Clustering (Müllner, 2011). Similar al resto de los módulos, se pueden utilizar las estrategias ya desarrolladas en librerías como scikit-learn (Pedregosa, et al., 2011) o SCiPY (Virtanen, et al., 2020).

La construcción de los tópicos depende en gran medida de este paso y la selección del algoritmo de agrupamiento. Dependiendo del algoritmo seleccionado se obtienen distintas variaciones como una clase adicional de datos atípicos, que se obtiene al usar HDBSCAN, o se puede especificar un número predefinido de tópicos, por ejemplo, con K-Means.

1.4.4 Tokenización

Es el primer paso en la representación de los tópicos, en este paso se crean los términos con los cuales será representado el tópico. Los términos pueden estar conformados por una o varias palabras, dependiendo de la configuración del modelo. Adicionalmente, este paso se puede seleccionar si se desea remover stop words o la frecuencia mínima de un término para ser considerado en la representación del tópico.

En BERTopic el algoritmo de tokenización por defecto es CountVectorizer (Scikit-learn developers, 2024), que forma parte de la librería scikit-learn. El algoritmo es usado para convertir una colección de documentos en una representación numérica matricial llamada matriz documento-término donde las filas representan documentos, las columnas representan token o palabras y valores números de cada celda representa la frecuencia del token *i* en el documento *j*.

Otra alternativa para obtener los pesos de los términos es el algoritmo Term Frequency – Inverse Document Frequency (TF-IDF) (Scikit-learn developers, 2024). En este algoritmo, similar a CountVectorizer, se obtiene la matriz documento-término, sin embargo, las frecuencias son ponderadas en base a la frecuencia del token en el documento con respecto a la frecuencia del token en todos los documentos.

$$tf - idf(t,d) = tf(t,d) \times \left(\log \frac{1+n}{1+df(t)} + 1\right)$$

Donde:

$$tf(t,d) = frecuencia\ del\ término\ t\ en\ el\ documento\ d$$
$$n = número\ total\ de\ documentos$$
$$df(t) = número\ de\ documentos\ que\ contienen\ el\ término\ t$$

Ecuación 1. Term Frequency – Inverse Document Frequency.
Fuente: *(Scikit-learn developers, 2024)*

1.4.5 Ponderado de pesos

Una de las características propias de BERTopic es la utilización de Class-based Term Frequency – Inverse Document Frequency (c-TF-IDF) (Grootendorst M., 2020), la cual es una versión ajustada del algoritmo Term Frequency – Inverse Document Frequency (TF-IDF). Esta versión funciona a nivel clase a diferencia de TF-IDF que funciona a nivel documento, esto se consigue generando un solo vector de clase al concatenar todos los documentos que conforman la clase.

$$c - TF - IDF_i = \frac{t_i}{w_i} \times \log \frac{m}{\sum_j^n t_j}$$

Donde:

$$t_i = frecuencia\ cada\ palabra\ en\ la\ clase\ i$$
$$w = número\ total\ de\ palabras$$
$$m = número\ de\ documentos$$
$$n = clases$$

Ecuación 2. Class-based Term Frequency – Inverse Document Frequency.
Fuente: *(Grootendorst M. , 2020)*

1.4.6 Representación de tópicos

Este paso final consiste en darle un nombre o significado a cada uno de los tópicos generados en el paso anterior. En este paso se puede generar tantas representaciones como se desee usando los modelos disponibles, entre los que se encuentran: KeyBERTInspired (Grootendorst M. P., 2022), PartOfSpeech (Grootendorst M. P., 2022), MaximalMarginalRelevance (Grootendorst M. P., 2022), entre otros. En BERTopic no hay una representación de tópicos por defecto y es un paso opcional.

1.5 Métricas de evaluación

Un paso fundamental de las metodologías de proyectos de minería de datos es la evaluación de los resultados obtenidos. En el caso del modelado de tópicos, al ser un modelo no supervisado resulta desafiante la obtención de métricas cuantitativas; sin embargo, en trabajos como Dieng, Ruiz, & Blei, (2020) y Grootendorst M. P. (2022) se propone el uso de las métricas coherencia y diversidad como indicador de la calidad de los tópicos obtenidos.

1.5.1 Coherencia

El objetivo de la métrica de coherencia es medir qué tan interpretables son los tópicos para los humanos (Mimno, Wallach, Talley, Leenders, & McCallum, 2011); los tópicos coherentes tendrán palabras que suelen ocurrir más frecuentemente en los mismos documentos.

Existen varias alternativas para el cálculo de la métrica de coherencia, cada una de ellas con variantes que buscan correlacionarse de mejor manera con la interpretabilidad humana.

Una de la alternativas es la coherencia UCI (C_{UCI}) (Röder, Both, & Hinneburg, 2015), la cual se basa en el cálculo Información Mutua Puntual (PMI). Se define como la media de la relación log_2 de la frecuencia de coocurrencia de las palabras w_i y w_j dentro de un tópico i (Rahimi, y otros, 2024).

$$C_{UCI} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} PMI(w_i, w_j)$$

Ecuación 3. Coherencia UCI.
Fuente: *(Harrando, Lisena, & Troncy, 2021)*

Donde PMI es la información mutua puntual:

$$PMI(w_i, w_j) = log \frac{P(w_i, w_j) + \epsilon}{P(w_i)\, P(w_j)}$$

$$N = número\ de\ tópicos$$

Ecuación 4. Información Mutua Puntual.
Fuente: *(Harrando, Lisena, & Troncy, 2021)*

(Mimno, Wallach, Talley, Leenders, & McCallum, 2011) proponen el cálculo de la coherencia UMass (C_{UMass}) como el número de veces que un par de palabras coocurren en el corpus y compara este número con el número esperado de coocurrencias cuando las palabras se distribuyen aleatoriamente por todo el corpus (Rahimi, y otros, 2024).

$$C_{UMass} = \frac{2}{N(N-1)} \sum_{i=2}^{N} \sum_{j=1}^{i-1} log \frac{P(w_i, w_j) + \epsilon}{P(w_j)}$$

Ecuación 5. Coherencia UMass.
Fuente: *(Harrando, Lisena, & Troncy, 2021)*

Otra alternativa es la utilización del cálculo de la Información Mutua Normalizada por Puntos (NPMI) (Bouma, 2009). NPMI tiene la ventaja de tomar en cuenta que algunas palabras son más frecuentes que otras en el corpus y ajusta las frecuencias de acuerdo con esto (Rahimi, y otros, 2024).

$$NPMI(w_i, w_j) = \frac{PMI(w_i, w_j)}{-\log\left(p(w_i, w_j) + \epsilon\right)}$$

Ecuación 6. Información Mutua Normalizada por Puntos.
Fuente: *(Bouma, 2009)*

Aunque existen varias implementaciones de coherencia utilizando NPMI (C_{NPMI}) la más frecuente es utilizando C_{UCI} con el reemplazo de PMI por NPMI (Harrando, Lisena, & Troncy, 2021).

$$C_{NPMI} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} NPMI(w_i, w_j)$$

Ecuación 7. Coherencia NPMI.
Fuente: *(Harrando, Lisena, & Troncy, 2021)*

1.5.2 Diversidad

La diversidad de tópicos se define como el porcentaje de palabras únicas en el top N de palabras de todos los tópicos (Dieng, Ruiz, & Blei, 2020). La métrica se mide de 0 a 1; donde 0 es una diversidad baja, es decir los tópicos son redundantes, y 1 que indica que los tópicos son variados.

La diversidad de acuerdo con (Dieng, Ruiz, & Blei, 2020) se define como:

$$Diversidad = \frac{W}{T \times k}$$

Donde:

$$W = Palabras\ únicas\ de\ todos\ tópicos$$
$$T = Número\ de\ tópicos$$
$$k = Principales\ k\ palabras\ del\ tópico$$

Ecuación 8. Diversidad.

Fuente: *(Dieng, Ruiz, & Blei, 2020)*

Idealmente al medir la diversidad de los tópicos generados nos gustaría obtener valores cercanos a 1, indicándonos que los tópicos generados no son repetitivos entre sí.

1.6 Optimización de modelos de tópicos

En general los modelos de aprendizaje automático requieren el ajuste de ciertos parámetros de entrada que definen el comportamiento, estructura y potencialmente los resultados del modelo. Para que un modelo de aprendizaje automático genere los mejores resultados es necesario ajustar estos parámetros de tal modo que se ajusten de la mejor manera posible al problema que está resolviendo. A la búsqueda de estos parámetros óptimos se le denomina optimización de hiperparámetros.

La optimización de hiperparámetros es un problema relativamente nuevo, y aunque los primeros trabajos al respecto se remontan a la década de 1990 (Feurer & Hutter, 2019), recientemente con el uso cada vez mayor de modelos de aprendizaje automático en todas las áreas de conocimiento, ha habido un crecimiento en el interés y desarrollo de metodologías de optimización.

Entre los principales métodos que se tienen hoy en día se encuentran: búsqueda por cuadrícula, búsqueda aleatoria, métodos basados en población y optimización bayesiana (Feurer & Hutter, 2019).

- Búsqueda por cuadrícula. En este método el usuario debe definir un conjunto finito de valores de búsqueda para cada hiperparámetro, y la búsqueda por cuadrícula evalúa el producto cartesiano de estos conjuntos; el problema de este método es que sufre por la dimensionalidad ya que al incrementar los parámetros el número de evaluaciones crece exponencialmente (Feurer & Hutter, 2019).

- Búsqueda aleatoria. Es una alternativa de la búsqueda por cuadrícula, como su nombre lo indica la búsqueda aleatoria prueba configuraciones generadas

a partir de una distribución de probabilidad conservando la mejor de las muestras.

- Métodos basados en población. Entre los que se encuentran los algoritmos genéticos, algoritmos evolutivos, estrategias evolutivas y la optimización por enjambre de partículas. Estos algoritmos funcionan generando una población o conjunto de posibles respuestas y aplican mutaciones o cruces entre miembros para obtener una nueva generación de mejores configuraciones (Feurer & Hutter, 2019).

- Optimización bayesiana. De acuerdo con (Feurer & Hutter, 2019) la optimización bayesiana se puede definir como un algoritmo iterativo conformado por: un modelo sustituto probabilístico y una función de adquisición para decidir el siguiente punto a evaluar. Con cada iteración el modelo sustituto es ajustado a todas las observaciones de la función objetivo hasta el momento. La función de adquisición, que utiliza la distribución predictiva del modelo probabilístico, determina la utilidad de distintos puntos candidatos para seleccionar el mejor candidato.

1.7 Optimización con Optuna

Optuna es una librería de Python de código abierto para la optimización de hiperparámetros desarrollada a partir del trabajo de (Akiba, Sano, Yanase, Ohta, & Koyama, 2019) que utiliza técnicas de optimización bayesiana. Las principales ventajas de herramienta respecto a otras similares son:

- El espacio de búsqueda es construido de manera dinámica; a diferencia de otras herramientas donde se debe definir un espacio de búsqueda estático, en Optuna es posible agregar lógica de programación para la construcción y modificación del mismo.

- Eficiencia en los algoritmos de muestreo y podado. Para la parte de muestreo, Optuna es capaz de utilizar muestreo relacional y muestreo

independiente. En cuanto al podado, se refiere a la terminación temprano de pruebas que no son prometedoras con el objetivo de ahorrar recursos, para este objetivo Optuna utiliza una variación del algoritmo “Asynchronous Successive Halving Algorithm” (Liam, y otros, 2018) en cual tiene la ventaja de funcionar en ambientes distribuidos.

Capítulo 2

Modelado de tópicos

Capítulo 2. Modelado de tópicos

2.1 Recolección de Datos

De acuerdo al reporte de Deloitte “La banca mexicana en números para el tercer trimestre del 2021” (Méndez, 2021) las 7 principales instituciones con cartera de consumo en México son los siguientes bancos: BBVA México (30%), Banamex (15%), Banorte (12%), Santander (12%), Banco Azteca (8%), HSBC (7%) y Scotiabank (4%). El estudio presentado a continuación se enfocará en estas 7 instituciones por ser las más grandes en México y en su conjunto representan el 88% de la cartera de consumo del país.

Para el estudio, se consideran los comentarios de la red X (anteriormente Twitter), por lo cual como primer paso se identificaron las cuentas de la red social X de los canales oficiales de comunicación para cada una de estas instituciones bancarias, siendo seleccionadas las siguientes:

Institución bancaria	Cuenta principal	Cuenta de atención a clientes
BBVA	@BBVA_Mex	@BBVARe_mx
CitiBanamex	@Citibanamex	@ContactoCitibmx
Banorte	@Banorte_mx	@BanorteEscucha
Santander	@SantanderMX	-
Banco Azteca	@BancoAzteca	@ BAztecaAyuda
HSBC	@HSBC_MX	-
Scotiabank	@ScotiabankMX	-

Cuadro 1. Cuentas de la red social X seleccionadas.

Fuente: Elaboración propia.

Para cada cuenta seleccionada se extrajo la totalidad de los tweets donde se etiquetaba dicha cuenta durante el periodo de recopilación de datos que comprende el periodo del 15 de octubre de 2022 al 4 de junio de 2023. La herramienta utilizada para la recolección fue la API v2.0 de Twitter para Python (X Corp., 2024) que permite a desarrolladores de código consultar y obtener menciones en Twitter y el historial de conversaciones públicas.

En total se extrajeron 113,634 tweets no únicos, dado que en un mismo tweet se podía hacer referencia a más de 1 cuenta, con la siguiente distribución entre cuentas:

Cuenta red social X	Número de tweets
@Citibanamex	15,022
@BancoAzteca	13,454
@BBVA_Mex	12,777
@BBVARe_mx	12,397
@SantanderMx	12,012
@HSBC_MX	11,524
@Banorte_mx	10,900
@ContactoCitibmx	9,146
@ScotiabankMX	6,367
@BanorteEscucha	5,694
@BAztecaAyuda	4,341

Cuadro 2. Distribución de tweets por cuenta.

Fuente: Elaboración propia.

Una vez con los tweets por cuenta se procedió a agrupar por banco, ya que no interesa para el objetivo de este análisis obtener métricas por cuenta de Twitter.

Adicional al texto que conforma un tweet, la API de Twitter v2 permite obtener datos complementarios asociados al tweet, los recopilados para este estudio fueron los siguientes:

Campo	Descripción [1]	% de registro no nulos
Id	Identificador único del tweet	100%
Author ID	Identificador único del usuario creador	100%
Text	Contenido del tweet	100%
Source	Aplicación o plataforma desde la cual el usuario generó el tweet	19%
Entities	Detalles del texto con significado especial en el Tweet (por ejemplo, etiquetas o hashtags).	100%
Created At	Fecha y hora de creación	100%

[1] Descripción del campo obtenida de la documentación oficial de la API de Twitter v2 (X Corp., 2024).

Lang	Idioma	100%
Referenced Tweets	Arreglo de Tweet a los cuales el Tweet actual hace referencia	63%
In Reply To Userid	Solo cuando el Tweet es una réplica. Identificador del usuario del Tweet original	83%
Edit History Tweet Ids	Arreglo de identificadores de Tweet para cada una las versiones anteriores del Tweet	100%
Conversation Id	Identificador del Tweet original de la conversación sobre el cuál se replicó	100%
Attachments	Tipo de adjunto	16%
Geo	Locación referida en el Tweet en caso de ser etiquetada en el Tweet.	3%

Cuadro 3. Detalle de campos obtenidos a través de la API Twitter v2.
Fuente: Elaboración propia.

Como se puede apreciar en el Cuadro *3*, los campos Attachments y Geo contienen muy poca información teniendo datos solamente 15% y 3% de los registros respectivamente. Por lo tanto, se tomó la decisión de excluir estos campos y enfocarse en aquellos que pudieran proporcionar más información.

2.2 Limpieza de comentarios

Al igual que cualquier otro modelo de aprendizaje automático, la calidad de los modelos de texto depende en gran medida de la calidad de los datos con los que son alimentados (Gudivada, Apon, & Ding, 2017). Dado que los comentarios recopilados pueden ser generados por diversos actores que no son de interés para este estudio, tales como empleados de las instituciones bancarias (agentes de atención a clientes o community managers encargados de administrar las redes sociales), resulta de suma importancia poder identificar los comentarios que pertenezcan exclusivamente a clientes o clientes potenciales.

Para filtrar los comentarios que no son de interés para el estudio se definieron un conjunto de reglas con base en razonamiento heurístico e identificación de patrones, las cuales son consistentes con lo observado en los datos y la manera en que los usuarios interactúan en la red social X.

En primera instancia se utilizó, para aquellos casos donde se contaba con el detalle, la información proporcionada por Twitter mismo respecto a la fuente de tweet (campo Source ver Cuadro *3*).

Analizando los datos se observa en el Cuadro *4*, que este atributo puede ser de mucha ayuda para identificar el tipo de autor, los Tweet generados a través de las aplicaciones de Twitter corresponden en su mayoría a clientes, mientras que los Tweets generados por herramientas o plataformas de gestión de redes sociales corresponden en su mayoría a agentes de atención a clientes.

Fuente	Número de tweets
Twitter for Android	8129
Twitter for iPhone	6381
Twitter Web App	2344
S1Gateway	969
TweetDeck	261
IFTTT	178
Salesforce - Social Studio	161
Otras fuentes	478

Cuadro 4. Número de tweets por fuente.
Fuente: Elaboración propia.

En segunda instancia, se llevó a cabo un filtro por número de tweets por autor. El concepto detrás de esta segmentación es que un agente de atención a clientes seguramente responderá muchos tweets todos los días mientras que un cliente mandará un número relativamente bajo de tweets y por periodos cortos de tiempo, una vez que la consulta del usuario haya sido resuelta ya no tendrá motivo para seguir interactuando.

En los datos se observaron autores con volúmenes muy altos de tweets, sin embargo, la cantidad de autores con este comportamiento es mínima Figura *4*. El valor de corte utilizado fue de 10 tweets, con este valor de corte de los 51,550 autores distintos fueron descartados 512 autores por tener comportamiento de agentes de atención al cliente. Cabe resaltar que si bien, estos representan el 1% de los autores, en su conjunto publicaron cerca del 15% tweets.

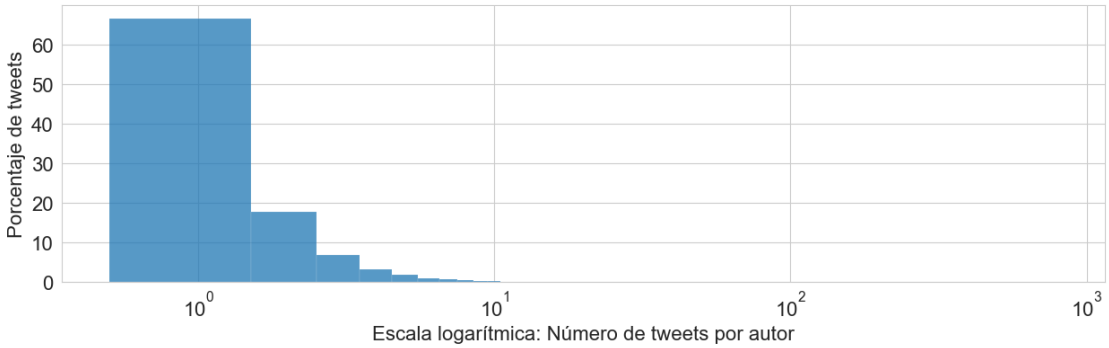


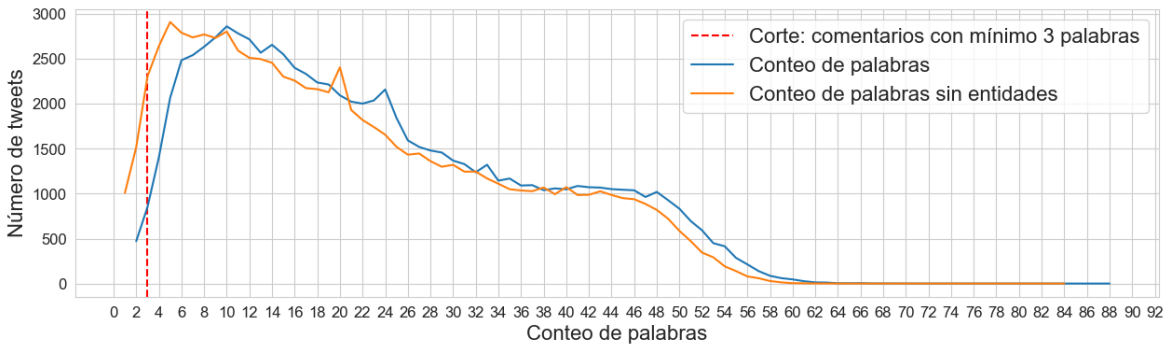
Figura 4. Número de tweets por autor.

Elaboración propia.

La última fase de la limpieza consistió en filtrar los tweets por longitud del texto eliminando aquellos que eran muy cortos. El conteo de palabras se llevó a cabo de 2 formas; en primera instancia un conteo simple de palabras, sin embargo, esto puede ser engañoso ya que muchos tweets cortos están formados en su mayor parte por entidades[2].

Por ejemplo, el tweet: *"@langielaw @FundacionAzteca @BancoAzteca @BAztecaAyuda @AztecaCEFAT @Azteca @AztecaEdoMex @AztecaDeportes @Est_Churubusco @jzarzap @carolina_rocha_ @Javier_Alatorre Gracias"*, por conteo simple está conformado de 13 palabras, no obstante, al remover las entidades solo se tiene una palabra.

Por tal motivo, se calculó el conteo de palabras sin entidades, que por ser la parte del texto donde los clientes describen su solicitud o comentario, es la que más interesa a este análisis. En la Figura *5* se puede observar esta diferencia entre ambos conteos.



[2] Entidades o Entities, conformado por elementos de texto con un significado especial en el tweet tales como: hashtags, menciones, URLs, anotaciones, cashtags, entre otros. (X Corp., 2024)

Figura 5. Comparación de número de tweets entre conteo de palabras simple y conteo de palabras sin entidades.

Elaboración propia.

Posterior a una revisión manual de tweets con diferente frecuencia de palabras y con el objetivo de descartar la menor cantidad posible de comentarios, se tomó la determinación de descartar únicamente aquellos tweets que tuvieran menos de 3 palabras sin considerar entidades.

El resultado final del proceso de limpieza significó reducir la base de tweets en un 25%, dejando un total de 82,190 tweets para los pasos subsecuentes.

2.3 Preparación de texto

Como cualquier otro modelo de aprendizaje automático, el modelo de tópicos requiere de una preparación previa de las variables de entrada, en este caso, el texto. Existen varias técnicas comunes para preparar el texto antes de ser ingresado al modelo, cuyos objetivos principales son reducir el ruido en los datos y disminuir el vocabulario. Las técnicas utilizadas para este estudio fueron:

1. Normalización Unicode del tipo Normalization Form KC (NFKC). Utilizado para filtrado de caracteres especiales, se utiliza una descomposición de compatibilidad, seguida de una composición canónica (Whistler, 2023). Se utilizó la librería unicodedata (Python Software Foundation, 2024). La normalización Unicode se utilizó para remover caracteres especiales, acentos y caracteres no validados o que carecen de significado para el análisis, tales como saltos de línea, tabuladores y caracteres que no pertenecen al idioma español.

   Ejemplo de normalización Unicode NFKC:

   Tweet original – *“@BBVARe_mx \nFui a la Reposición de un una tarjeta de crédito y débito.\nEl personal en ningún momento me ayudó a desbloquear mi aplicación y me sacaron que ya era tarde.\nSeñores si su gente no quiere atender pues corta los.\nY de verdad hagan algo ustedes.\nhttps://t.co/MBFpcEqraj”*

   Tweet normalizado – *“bbvare_mx fui a la reposicion de un una tarjeta de credito y debito el personal en ningun momento me ayudo a desbloquear mi aplicacion y me*

sacaron que ya era tarde señores si su gente no quiere atender pues corta los y de verdad hagan algo ustedes httpstcombfpceqraj".

2. Remover entidades de tipo hashtag (#), URL (http y https) y menciones (@). Para ello se utilizaron los metadatos asociados a cada tweet que la API de Twitter proporciona.

3. Emojis. Sustitución de emojis por su significado en texto, utilizando la librería emoji 2.10.0 (Kim & Wurster, 2024).

   Tweet original – *"@Sonia_2601 @BBVARe_mx A mi ya y @bbva no me da respuesta, me dijeron que era mentira lo que estaba diciendo porque en su "sistema" no aparecen los movimientos, ya me jalaron $4,000 y ahora nuevamente $10,000 pero claro que iré a @CondusefMX 😭 ".*

   Tweet con sustitución de emojis – *"A mi ya y no me da respuesta, me dijeron que era mentira lo que estaba diciendo porque en su "sistema" no aparecen los movimientos, ya me jalaron $4,000 y ahora nuevamente $10,000 pero claro que iré a cara_llorando_fuerte".*

4. Dinero. Reemplazo de las cantidades numéricas que hace referencia a montos de dinero por la palabra "dinero". Para ello se usaron expresiones regulares.

5. Remover stop words o palabras vacías. Son palabras que ocurren frecuentemente en el texto pero que no proporcionan información adicional o valiosa del contenido del mismo. Algunos ejemplos de stop words en el idioma español son los artículos (un, una, el, la, etc.) y preposiciones (a, ante, bajo con, de, desde, etc.). Para ello se utilizó la lista de stop words para el idioma español proporcionada por la librería nltk 3.8.1 (Bird, Klein, & Loper, 2009).

   Ejemplo de eliminación de stop words:
   Tweet original: *"@BBVARe_mx @BBVA_Mex Me hicieron una transferencia desde hace dos horas y aún NO se refleja en mi saldo. La clave de rastreo Banxico marca la operación como liquidada. https://t.co/lC7AyPzzFQ"*

Tweets con eliminación: *“@BBVARe_mx @BBVA_Mex Me hicieron transferencia hace dos horas aún NO refleja saldo. La clave rastreo Banxico marca operación liquidada. https://t.co/lC7AyPzzFQ”.*

6. Signos de interrogación y exclamación. Con el objetivo de conservar el contexto del texto respecto a si se trata de una pregunta o un texto enfatizado con signos de exclamación se realizó una sustitución de los mismos por las palabras “interrogación” y “exclamación”.

   Tweet original: *“@BBVARe_mx ¿pueden ayudarme con un movimiento en tránsito? Hice una transferencia para un pago y aun no se libera y tampoco tengo el dinero disponible en mi cuenta!!”*
   Tweet con sustitución: *“BBVARe_mx interrogacion pueden ayudarme con un movimiento en tránsito interrogacion Hice una transferencia para un pago y aun no se libera y tampoco tengo el dinero disponible en mi cuenta exclamacion”.*

7. Remover repeticiones de más de 2 caracteres iguales usando expresiones regulares.

Ejemplo de visualización de vocabulario del antes y después de la preparación y limpieza del texto:

Figura 6. Comparación de vocabulario, antes y después preparación de texto.

Elaboración propia.

Con el objetivo de experimentar como la utilización de métodos de preparación de texto afectan el resultado final del modelado de tópicos se creó la clase que implementa los pasos de preprocesamiento descrito, la cual puede consultarse en el anexo A.

Este diseño permitió prender y apagar cada uno de los métodos de manera independiente. De esta forma se obtiene la flexibilidad para preparar el texto con los métodos de interés y poder combinarlos entre ellos. Esto fue de especial utilidad en el paso posterior de optimización del modelo de tópicos.

La limpieza y preparación del texto permitió reducir el número de palabras en el vocabulario, sin modificar el contexto del mismo, reduciendo repeticiones y diferencias en la escritura y removiendo términos innecesarios. Por otra parte, a elementos como emojis y signos de exclamación e interrogación se les dio una representación que puede ser usada por los modelos de texto.

La diferencia entre vocabulario inicial y vocabulario procesado fue la mostrada en el Cuadro *5*.

	Tweets iniciales	Tweets procesados
Total, de documentos	82190	82190
Palabras distintas	140367	76794
Palabras distintas sin entidades	97638	42922
Promedio de palabras por documento	23.8	14.1
Promedio de palabras por documento sin entidades	21.5	11.2

Cuadro 5. Comparativa antes y después de procesamiento de texto.
Elaboración propia.

2.4 BERTopic

El primer paso en el desarrollo del modelo de tópicos fue la creación de un pipeline de Scikit-learn, el cual se puede consultar en el anexo B, para permitir de esta forma la ejecución secuencial del paso de preprocesamiento de datos seguido del modelo BERTopic. El pipeline se construyó de modo que cada entrada fuera una variable que se pudiera modificar para crear versiones alternas del mismo pipeline, cada una

con una configuración de algoritmos e hiperparámetros distintos, y de este modo poder comparar estas alternativas con el objetivo de encontrar aquellas con el mejor desempeño en las métricas de evaluación seleccionadas. Estas variables serán las que conformen el espacio de búsqueda durante la fase de optimización.

Como se muestra en el Cuadro *6*, para la versión inicial del modelo se utilizaron todos los pasos de preprocesamiento, así como los valores por defecto de BERTopic.

Paso pipeline	Parámetro	Valor por defecto
Preprocesamiento	normalizar	True
Preprocesamiento	emojis	True
Preprocesamiento	acentos	True
Preprocesamiento	dinero	True
Preprocesamiento	caracteres especiales	True
Preprocesamiento	repetición caracteres	True
Preprocesamiento	stop words	True
BERTopic	Modelo embedding	all-MiniLM-L6-v2
BERTopic	Modelo reducción dimensional	UMAP(n_neighbors=15, n_components=5, min_dist=0.0, metric='cosine')
BERTopic	Modelo agrupamiento	HDBSCAN(min_cluster_size=15, metric='euclidean', cluster_selection_method='eom', prediction_data=True)
BERTopic	Modelo tokenización	CountVectorizer(min_df=10)
BERTopic	Modelo ponderado de pesos	ClassTfidfTransformer()
BERTopic	Representación de tópicos	KeyBERTInspired()
BERTopic	Idioma	multilingual

Cuadro 6. Configuración de modelo inicial BERTopic.
Elaboración propia.

Con respecto a la medición del desempeño de los modelos de tópicos se determinó utilizar las métricas coherencia y diversidad. Su uso combinado ofrece una forma objetiva y cuantitativa, si bien no perfecta, de medir la calidad de los resultados.

Como en el cálculo de la coherencia es necesario utilizar los tokens con los que se construyó el modelo de tópicos, resulta importante utilizar los mismos pasos de

preprocesamiento y el modelo de reducción dimensional. Para este objetivo se extrajeron los modelos del pipeline utilizado y con estos modelos se transformaron nuevamente los datos de entrada.

Para el cálculo de la métrica de coherencia se utilizó la librería de Python octis versión 1.13.1 (Terragni, Fersini, Galuzzi, Tropeano, & Candelieri, 2021), la cual implementa varios métodos de cálculo tales como: UMass, Coherence Vector (CV), UCI y Normalized Pointwise Mutual Information (NPMI). Con base en las pruebas realizadas se decidió utilizar el cálculo de coherencia UMass, tanto por ser uno de los métodos que mejor consistencia ofrecía como por su mayor velocidad de cálculo y menor uso de memoria con respecto a los otros. La implementación del método para el cálculo de la coherencia UMass puede ser consultado en el anexo C.

La métrica de diversidad resulta más sencilla de calcular debido a que solo requiere el uso de las palabras representativas de cada tópico. Al igual que para el cálculo de la coherencia, se utilizó la librería de Python OCTIS versión 1.13.1 (Terragni, Fersini, Galuzzi, Tropeano, & Candelieri, 2021). La implementación del método para el cálculo de la métrica diversidad puede ser consultado en el anexo D.

Como puntos de referencia se decidió utilizar los modelos LDA y NFM, así como una versión base de BERTopic que utilizase únicamente los valores por defecto. Se obtuvieron los resultados presentados en el Cuadro 7. Como se puede observar, la implementación por defecto del modelo BERTopic, resulta inferior a los algoritmos LDA y NFM para ambas métricas de evaluación; siendo NFM el algoritmo con los mejores valores de coherencia y LDA el algoritmo con la mejor diversidad.

Modelo	Coherencia UMass	Coherencia UCI	Coherencia NPMI	Diversidad
BERTopic inicial	-11.36	-5.95	-0.19	0.43
LDA	-5.11	-0.83	0.04	**0.70**
NFM	**-2.92**	**0.34**	**0.07**	0.65

Cuadro 7. Métricas de desempeño de los modelos de tópicos base.
Elaboración propia.

Capítulo 3

Selección de algoritmos y optimización de hiperparámetros en BERTopic

Capítulo 3. Selección de algoritmos y optimización de hiperparámetros en BERTopic

Al ser BERTopic una estructura modular conformada por un conjunto de algoritmos, existe la posibilidad de optimizarlo ya sea mediante la selección óptima de los algoritmos que lo conforman o mediante la hiperparametrización de cada uno de estos algoritmos. Es por esto que la propuesta de este trabajo es generar una optimización por etapas, donde la primer etapa consiste en seleccionar la combinación modelos (embedding, reducción dimensional y agrupamiento) con el mejor desempeño y la segunda etapa la optimización de los hiperparámetros de los modelos ganadores de la primer etapa junto con el resto de los hiperparámetros.

Para ambas etapas de entrenamiento se utilizó la librería Optuna versión 3.4.0 (Akiba, Sano, Yanase, Ohta, & Koyama, 2019) la cual realiza un proceso de optimización bayesiana, así como varias técnicas enfocadas en la mejora de desempeño y convergencia.

3.1 Optimización para selección de algoritmos

Aunque el objetivo principal de esta primera etapa de selección de algoritmos es identificar los mejores modelos para cada uno de los módulos de BERTopic, también se agregaron algunos parámetros propios de cada modelo con el objetivo de darle cierta flexibilidad a estos modelos y no limitarlos a una configuración única.

Los hiperparámetros seleccionados a optimizar en la primera etapa fueron los siguientes:

Paso pipeline	Parámetro	Espacio de búsqueda
Preprocesamiento	normalizar	True, False
Preprocesamiento	emojis	True, False
Preprocesamiento	dinero	True, False
Preprocesamiento	stop words	True, False
BERTopic: Embedding	Modelo embedding	paraphrase-multilingual-MiniLM-L12-v2, paraphrase-multilingual-mpnet-base-v2,

		distiluse-base-multilingual-cased-v1, nlp, hiiamsid/sentence_similarity_spanish_es universal-sentence-encoder-multilingual
BERTopic: Reducción dimensional	Modelo reducción dimensional	UMAP, PCA, KMeans
BERTopic: Reducción dimensional	umap_model_n_neighbors	Número entero: [5, 20]
BERTopic: Reducción dimensional	umap_model_n_components	Número entero: [2, 10]
BERTopic: Reducción dimensional	umap_model_metric	euclidean, cosine
BERTopic: Reducción dimensional	umap_model_n_clusters	Número entero: [2, 10]
BERTopic: Agrupamiento	Modelo clusterización	HDBSCAN, KMeans
BERTopic: Agrupamiento	cluster_model_min_cluster_size	Número entero: [10, 50]
BERTopic: Agrupamiento	cluster_model_n_clusters	Número entero: [3, 10]
BERTopic: Ponderado pesos	vectorizer_min_df	Número decimal: [0, 1]
BERTopic: Ponderado pesos	vectorizer_ngram_range	[1,1], [1,2], [1,3]
BERTopic: Tokenización	ctfidf_bm25_weighting	True, False
BERTopic: Tokenización	ctfidf_red_freq_words	True, False
BERTopic	language	Multilingual, Spanish

Cuadro 8. Parámetros a optimizar etapa 1.
Elaboración propia.

En esta primera etapa de optimización se ejecutaron 1000 iteraciones, es decir, el proceso de optimización en Optuna probó 1000 configuraciones diferentes para encontrar la mejor posible.

Aunque la base de tweets para entrenamiento contenía 82190 tweets, dado el volumen de los datos y el alto costo computacional asociado a la construcción del modelos de tópicos se tomó la determinación de entrenar esta etapa con muestras

aleatorias de 4000 tweets donde para para ejecución se obtuvo una muestra aleatoria distinta.

3.1.1 Resultados optimización etapa 1

Durante el proceso de optimización se observó una mejora consistente y continua hacia valores más altos de diversidad y coherencia. Como se observa en la Figura *7* el proceso de optimización desde iteraciones tempranas obtuvo modelos con valores altos de diversidad, de hecho, se generaron bastantes modelos con diversidad igual a 1, sin embargo, la optimización de la coherencia requirió de más iteraciones donde se generaron modelos con valores más dispersos.

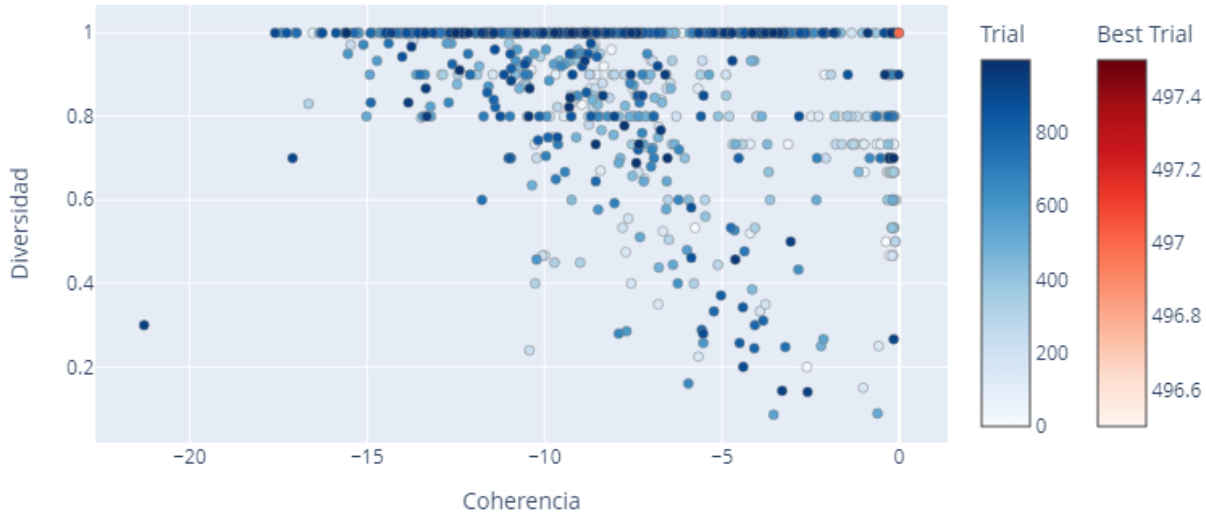


Figura 7. Resultados optimización etapa 1. Coherencia versus Diversidad.

Elaboración propia.

En los resultados obtenidos se observó que, de las 1000 iteraciones, 130 de estas terminaron con estado fallido (FAIL). Esto significa que el estudio de Optuna, equivalente a una ejecución de BERTopic, concluyó con error o no se generaron resultados, es decir, no se encontraron tópicos con la selección de parámetros del estudio.

Al observar la relación entre los hiperparámetros y las iteraciones fallidas se identificó una relación proporcional entre el hiperparámetro *vectorizer_min_df* y el número de ensayos terminados con estado fallido, como se observa en la Figura *8*. Esto se explica debido a que para la clase *CountVectorizer* utilizada en el paso de tokenización de tópicos, el parámetro *min_df* define un límite que descarta las

palabras que tienen una frecuencia en documentos menor al valor establecido. A medida que este valor incrementa, el modelo se vuelve más estricto dificultando la creación de tópicos.

Se tomó la decisión de limitar el parámetro *min_df* a un máximo de 0.65, que es aproximadamente el valor a partir del cual el número de ensayos fallidos empieza a incrementar considerablemente, esto de cara a la optimización de la etapa 2 con el objetivo de evitar ensayos fallidos.

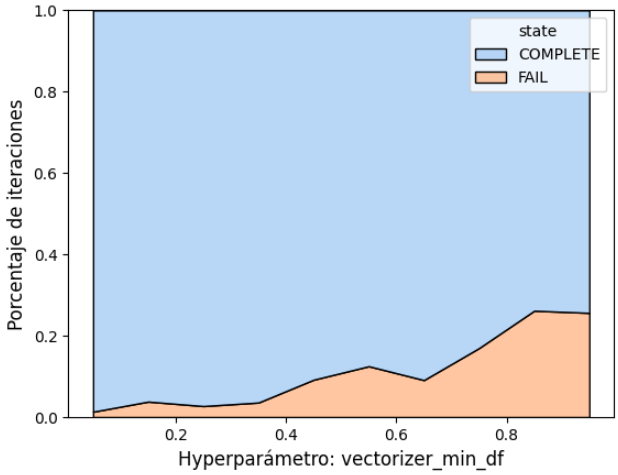


Figura 8. Iteraciones con error versus vectorizer_min_df.

Elaboración propia.

Resulta de interés el análisis posterior de la importancia de los hiperparámetros en el proceso de optimización Figura *9* y Figura *10*. Para el cálculo de la importancia se utilizó el método fANOVA (Hutter, Hoos, & Leyton-Brown, 2014) el cuál utiliza un modelo de regresión de bosque aleatorio para predecir los valores objetivo de los estudios terminados con estatus “COMPLETE”, es decir no fallidos, con base en la configuración de los hiperparámetros propios del estudio.

En la gráfica de importancia de la métrica de coherencia Figura *9* aparece en primer lugar la variable *vectorizer_min_df* que como se describió anteriormente, impacta en gran medida el resultado final del modelo. Las variables de interés en la etapa 1 de optimización *embedding_model*, *umap_model* y *cluster_model* aparecen, por orden de importancia, en las posiciones 2, 4 y 6, respectivamente.

Que estas variables tengan una importancia alta resulta un punto a favor de la optimización por etapas dado que el principal interés de esta primera etapa consiste

en encontrar la combinación algoritmos de embedding (*embedding_model*), reducción dimensional (*umap_model*) y agrupamiento (*cluster_model*) que serán utilizaron para la subsecuente etapa.

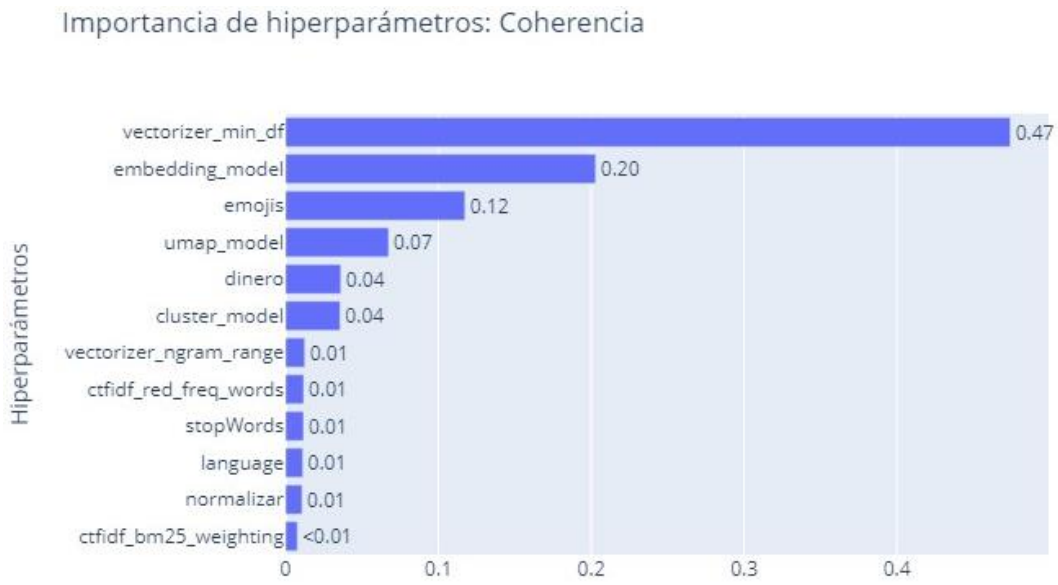


Figura 9. Resultados optimización etapa 1. Importancia de hiperparámetros para la métrica coherencia.

Elaboración propia.

En el caso de importancia de la métrica de diversidad Figura *10*, se da un caso similar al anterior, donde la variable con mayor peso resultó ser *vectorizer_min_df*. Las variables *embedding_model*, *umap_model* y *cluster_model* se encuentran en las posiciones 3, 4 y 8 respectivamente. Aunque la importancia de estas 3 variables de interés no fue tan alta como se esperaba, el resultado no es desalentador al considerar que para el caso de la diversidad al proceso de optimización le fue sencillo encontrar valores altos, cercanos a 1, desde un principio.

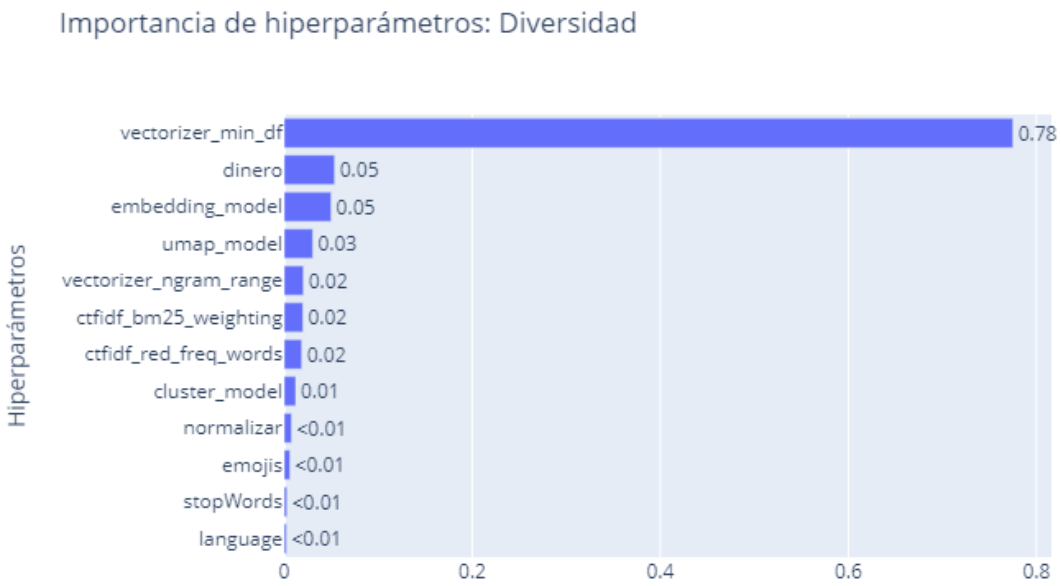


Figura 10. Resultados optimización etapa 1. Importancia de hiperparámetros para diversidad.

Elaboración propia.

El proceso de optimización de la etapa 1 se ejecutó en un tiempo aproximado de 32 horas, en las cuales se realizaron 1000 ciclos de entrenamiento con 4000 tweets cada uno. Se utilizó un procesador 13th Gen Intel Core i9-139000HX de 24 núcleos y una tarjeta de gráficos NVIDIA GeForce RTX 4080.

3.1.2 Selección de configuración ganadora etapa 2

Como se observa en la Figura *11* a los resultados de la métrica coherencia se les aplicó una transformación de estandarización y escalado, con el objetivo de obtener una métrica final que se encontrase en el intervalo [0,1].

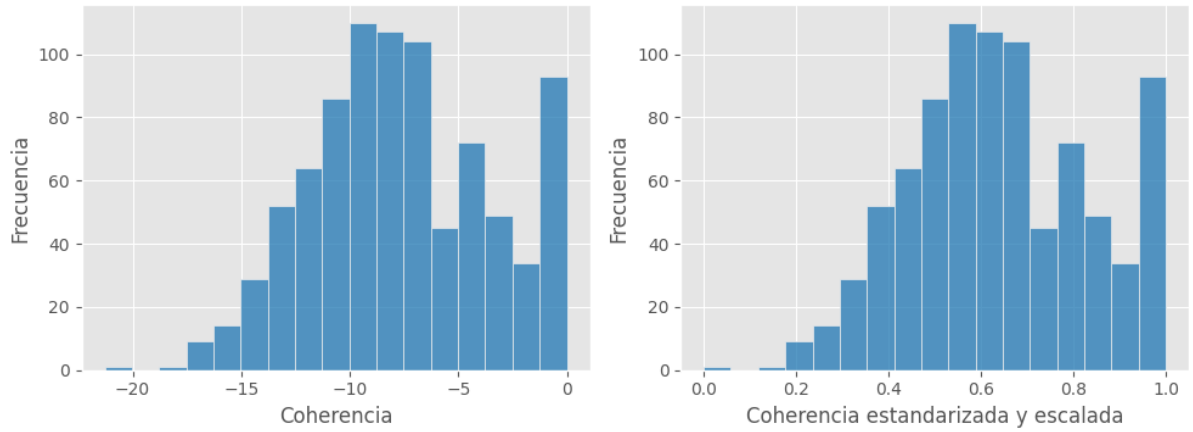


Figura 11. Histograma coherencia y coherencia estandarizada y escalada.

Elaboración propia.

En el caso de la métrica diversidad Figura *12*, la propia métrica está limitada al intervalo [0,1] por lo que no fue necesario realizar ninguna transformación adicional.

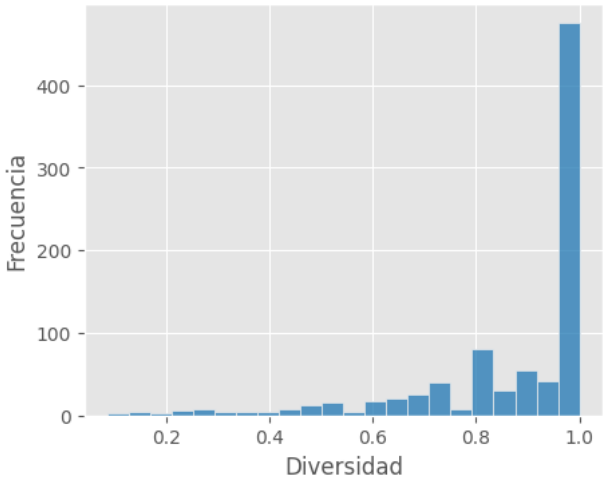


Figura 12. Histograma diversidad.

Elaboración propia.

Para la comparación entre modelos y la selección de la configuración con mejor desempeño se determinó otorgar la misma ponderación a ambas métricas, coherencia y diversidad, por lo que los modelos ganadores se obtuvieron a partir de los valores más altos de la media simple entre coherencia y diversidad.

Los 10 mejores modelos se muestran a continuación:

Hiperparámetros							
Embedding	**Reducción dimensional**		**Agrupamiento**		**Ponderado pesos**		
Modelo	**Modelo**	**Distancia/ Clústers/ Vecinos**	**Modelo**	**Tamaño mínimo**	**min_df/ ngram**	**Coherencia**	**Diversidad**
paraphrase-multilingual-MiniLM-L12-v2	UMAP	cosine 5 5	HDBSCAN	20	0.87 (1,3)	1.000000	1
paraphrase-multilingual-MiniLM-L12-v2	UMAP	cosine 5 7	HDBSCAN	20	0.89 (1,3)	0.999452	1
distiluse-base-multilingual-cased-v1	UMAP	cosine 5 5	HDBSCAN	20	0.92 (1,3)	0.999434	1
paraphrase-multilingual-MiniLM-L12-v2	UMAP	cosine 5 5	HDBSCAN	20	0.94 (1,3)	0.997797	1

paraphrase-multilingual-MiniLM-L12-v2	UMAP	cosine 5 5	HDBSCAN	10	0.71 (1,2)	0.996476	1
distiluse-base-multilingual-cased-v1	UMAP	euclidean 6 7	HDBSCAN	20	0.92 (1,3)	0.995648	1
paraphrase-multilingual-MiniLM-L12-v2	UMAP	cosine 3 9	HDBSCAN	10	0.55 (1,2)	0.993270	1
paraphrase-multilingual-MiniLM-L12-v2	UMAP	cosine 3 17	HDBSCAN	20	0.69 (1,3)	0.992879	1
paraphrase-multilingual-MiniLM-L12-v2	UMAP	cosine 7 5	HDBSCAN	10	0.93 (1,3)	0.991733	1
paraphrase-multilingual-mpnet-base-v2	KMeans	N/A 8 N/A	HDBSCAN	30	0.56 (1,3)	0.989763	1

Cuadro 9. Optimización etapa 1. Selección de los 10 mejores modelos.

Elaboración propia.

De los algoritmos para embedding probados, el algoritmo paraphrase-multilingual-MiniLM-L12-v2, fue el que se comportó de mejor manera siendo utilizado no solo en el modelo con mejor desempeño, sino también en 8 de las 10 mejores configuraciones. De forma similar, el algoritmo UMAP en el paso de reducción dimensional fue aquel con el mejor desempeño, como se muestra en el Cuadro *9* apareciendo en las 9 mejores configuraciones, superando a los algoritmos PCA y KMeans. En cuanto al algoritmo de agrupamiento, HDBSCAN obtuvo el mejor desempeño superando por mucho al algoritmo KMeans.

Finalmente, la configuración ganadora de la etapa 1, fue la siguiente:

Hiperparámetro	Configuración ganadora
Modelo embedding	paraphrase-multilingual-MiniLM-L12-v2
Modelo reducción dimensional	UMAP
Modelo agrupamiento	HDBSCAN
Idioma	Spanish
vectorizer_min_df	[0, 0.65]

Cuadro 10. Configuración ganadora optimización etapa 1.

Elaboración propia.

La configuración mostrada en el Cuadro *10* se mantendrá fija para la siguiente etapa, en la que se realizará la de búsqueda de hiperparámetros de cada uno de los modelos seleccionados en la primera fase.

3.2 Optimización de hiperparámetros de modelos seleccionados

Como se mencionó anteriormente, el objetivo de esta segunda etapa de optimización es realizar una búsqueda más fina de los valores óptimos para los hiperparámetro de los modelos ganadores de la etapa de optimización 1. El resultado esperado de esta segunda etapa es el modelo ganador con sus parámetros optimizados.

Los hiperparámetros a buscar en esta etapa fueron los siguientes:

Paso pipeline	Parámetro	Espacio de búsqueda
Preprocesamiento	normalizar	True, False
Preprocesamiento	emojis	True, False
Preprocesamiento	dinero	True, False
Preprocesamiento	stop words	True, False
BERTopic: Reducción dimensional	umap_model_n_neighbors	Número entero: [5, 20]
BERTopic: Reducción dimensional	umap_model_n_components	Número entero: [2, 10]
BERTopic: Reducción dimensional	umap_model_metric	euclidean, cosine
BERTopic: Agrupamiento	cluster_model_min_cluster_size	Número entero: [10, 50]
BERTopic: Ponderado pesos	vectorizer_min_df	Número decimal: [0, 0.65]
BERTopic: Ponderado pesos	vectorizer_ngram_range	[1,1], [1,2], [1,3]
BERTopic: Tokenización	ctfidf_bm25_weighting	True, False
BERTopic: Tokenización	ctfidf_red_freq_words	True, False

Cuadro 11. Parámetros a optimizar etapa 1.

Elaboración propia.

Cabe resaltar que si bien, los parámetros a optimización en esta etapa también fueron evaluados durante la etapa 1 de optimización, en este caso el objetivo final si es el de obtener la configuración final de estos los hiperparámetros.

En esta segunda etapa se realizaron 200 iteraciones de entrenamiento, sin embargo, en esta etapa para obtener el detalle fino esperado se determinó utilizar la base completa de 82190 tweets para cada iteración de entrenamiento.

3.2.1 Resultados optimización etapa 2

Al entrenar con la base de datos completa, se observa que la complejidad el modelo crece y por ende no se alcanzaron valores tan altos de coherencia y diversidad como los visto en la etapa de entrenamiento 1, sin embargo, en la Figura *13* si se aprecia que a medida que el proceso de entrenamiento avanzaba se fueron encontrando mejoras en ambas métricas, para en las últimas etapas del entrenamiento encontrar un punto de equilibrio entre ambas métricas (puntos de color rojo en la parte central de la gráfica).

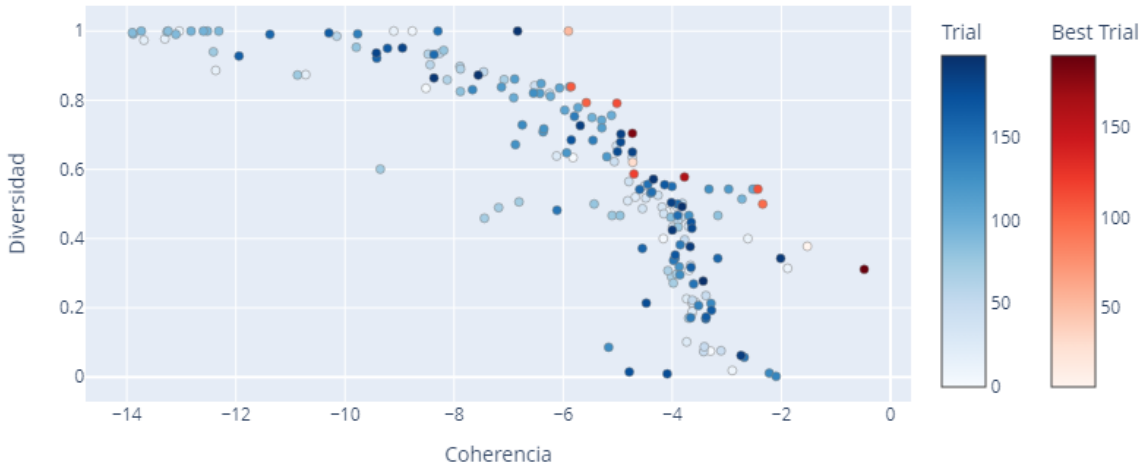


Figura 13. Resultados optimización etapa 2. Coherencia versus Diversidad.

Elaboración propia.

Para el cálculo de importancia de los hiperparámetros se utilizó de nuevo el método fANOVA explicado anteriormente. Como se observa en las Figura *14* y Figura *15*, se obtuvo un resultado similar a la optimización de la etapa 1, donde la variable

vectorizer_min_df nuevamente fue la que tuvo mayor importancia para el proceso de optimización para ambas métricas de desempeño.

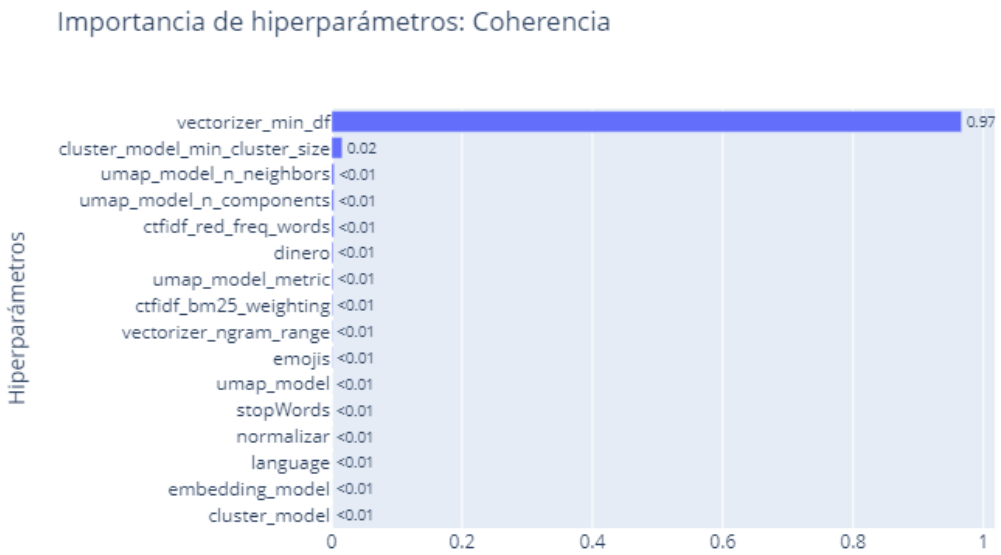


Figura 14. Resultados optimización etapa 2. Importancia de hiperparámetros para coherencia.

Elaboración propia.

En el caso de la métrica diversidad Figura *15* se observa que también fue de importancia la variable *cluster_model_min_cluster_size*, la cual indica el tamaño mínimo de los agrupamientos creados por el modelo HBDSCAN.

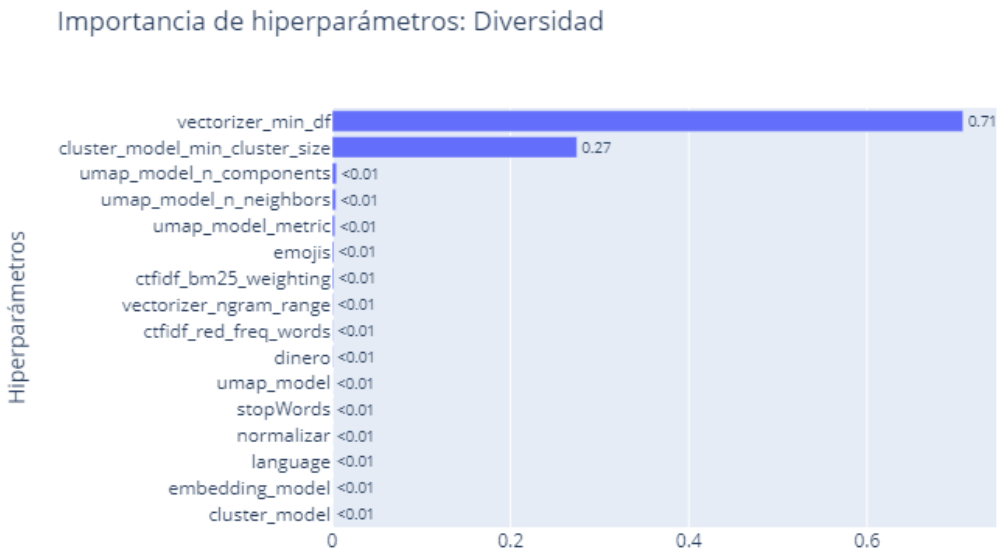


Figura 15. Resultados optimización etapa 2. Importancia de hiperparámetros para diversidad.

Elaboración propia.

El proceso de optimización de esta segunda etapa ejecutó en 3 horas y 46 minutos, lo cual fue considerablemente más rápido que el tiempo de ejecución de la etapa 1,

esto a pesar de haber sido efectuado con la totalidad de la base de tweets. Esta diferencia se explica debido a que el modelo de embeddings ganador de la etapa 1, paraphrase-multilingual-MiniLM-L12-v2, que deriva de Sentence-BERT (SBERT), es considerablemente más rápido que el resto de los modelos de embedding probados en la etapa anterior.

El desarrollo de SBERT se realizó a partir del trabajo de (Reimers & Gurevych, 2019) donde tomaron como base un modelo pre-entrenado BERT, y lo optimizaron para que fuese capaz de comparar utilizando la similitud coseno, de este modo se obtuvo una importante mejora en la reducción de tiempo de entrenamiento.

El hardware utilizado en esta segunda etapa de optimización fue el mismo que el utilizado en la etapa anterior.

3.2.2 Selección de configuración ganadora etapa 2

Del mismo modo a lo realizado en la etapa anterior, se aplicó la estandarización y escalado de la coherencia Figura *16*, obteniendo se así valores para ambas métricas en el intervalo [0,1].

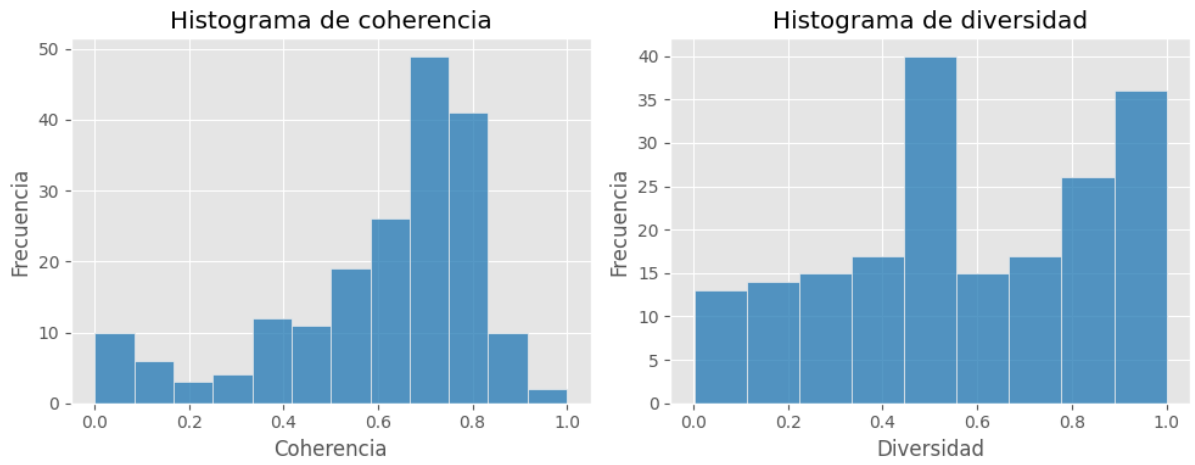


Figura 16. Histograma de coherencia estandarizada y escala, histograma de diversidad.

Elaboración propia.

Con una ponderación de importancia igual entre ambas métricas de desempeño, se obtuvieron los siguientes mejores modelos.

Hiperparámetros											
Clúster min	**Pesado bm25**	**Reducir palabras Frec.**	**Dinero**	**Emojis**	**Normalizar**	**Stop Words**	**Umap métrica**	**Umap componentes**	**Umap vecinos**	**Vect. min df**	**Vect. ngram range**

30	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	euclidean	4	17	0.24	3
50	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	euclidean	5	11	0.36	3
50	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	euclidean	8	9	0.27	3
50	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	euclidean	4	11	0.25	3
50	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	cosine	6	17	0.45	2
50	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	euclidean	8	9	0.29	3
50	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	euclidean	8	11	0.26	3
50	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	euclidean	5	11	0.25	3
50	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	euclidean	3	9	0.28	3
50	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	euclidean	3	9	0.29	3

Iteración	Coherencia	Diversidad
196	0.5263	1
112	0.662	0.7915
106	0.5994	0.8392
109	0.5836	0.8353
150	0.4174	1
103	0.6203	0.7931
111	0.5585	0.848
110	0.8547	0.5429
104	0.8478	0.5429
107	0.6088	0.7789

Cuadro 12. Optimización etapa 2. Selección de 10 mejores modelos.
Elaboración propia.

Finalmente, la configuración completa ganadora se obtuvo de evaluar los 10 mejores modelos. Para el caso de los valores numéricos se calculó la media de los valores obtenidos, para el caso de los valores categóricos se seleccionó aquel que tuviera la mayor frecuencia.

El resultado de esta selección se muestra en Cuadro *13*, donde los modelos de embedding, reducción dimensional y agrupamiento corresponde a los ganadores obtenidos en la *Etapa 1* y los hiperparámetros de estos modelos corresponden a las configuraciones obtenidas en la *Etapa 2*.

En el Cuadro *13* se observa de los métodos de preprocesamiento el proceso de optimización seleccionó: el uso de: reemplazo de dinero (convertir el carácter $ por la palabra dinero), reemplazo de emojis (convertir emojis por su significado en texto) y remover stop words.

En contraste, la normalización Unicode de texto no fue seleccionada. Esto se explica en parte debido a que los modelos transformadores de oraciones, como los usados por BERTopic para el paso de embedding, son entrenados con datos provenientes de interacciones humanas y por lo tanto funcionan mejor con texto del mismo tipo, sin una manipulación excesiva en el preprocesamiento.

Etapa Optimización	Paso pipeline	Hiperparámetro	Configuración ganadora
Etapa 1	BERTopic: Embedding	Modelo embedding	paraphrase-multilingual-MiniLM-L12-v2
Etapa 1	BERTopic: Reducción dimensional	Modelo reducción dimensional	UMAP
Etapa 1	BERTopic: Agrupamiento	Modelo agrupamiento	HDBSCAN
Etapa 1	BERTopic	Idioma	Spanish
Etapa 2	Preprocesamiento	dinero	True
Etapa 2	Preprocesamiento	emojis	True
Etapa 2	Preprocesamiento	normalizar	False
Etapa 2	Preprocesamiento	stop words	True
Etapa 2	BERTopic: Reducción dimensional	umap_model_metric	euclidean
Etapa 2	BERTopic: Reducción dimensional	umap_model_n_components	5
Etapa 2	BERTopic: Reducción dimensional	umap_model_n_neighbors	11
Etapa 2	BERTopic: Agrupamiento	cluster_model_min_cluster_size	50
Etapa 2	BERTopic: Ponderado pesos	vectorizer_min_df	0.3
Etapa 2	BERTopic: Ponderado pesos	vectorizer_ngram_range	[1,3]
Etapa 2	BERTopic: Tokenización	ctfidf_bm25_weighting	False
Etapa 2	BERTopic:	ctfidf_red_freq_words	False

Tokenización		

Cuadro 13. Configuración ganadora optimización etapa 2.

Elaboración propia.

Capítulo 4

Resultados

Capítulo 4. Resultados

Con los ajustes finales de la etapa 2 de optimización se reentrenó el modelo con la totalidad de los datos y se evaluaron las métricas de desempeño: Coherencia UMass, Coherencia UCI, Coherencia NPMI y Diversidad; obteniéndose los resultados mostrados en el Cuadro *14*.

Modelo	Coherencia UMass	Coherencia UCI	Coherencia NPMI	Diversidad
BERTopic inicial	-11.36	-5.95	-0.19	0.43
LDA (modelo base)	-5.11	-0.83	0.04	**0.70**
NFM (modelo base)	**-2.92**	**0.34**	**0.07**	0.65
BERTopic optimizado	-5.46	-1.21	0.01	0.67

Cuadro 14. Evaluación final de métricas de desempeño.
Elaboración propia.

En los resultados se observa una mejora considerable en todas las métricas de desempeño de la evaluación del modelo BERTopic inicial con respecto al modelo BERTopic optimizado. La optimización mediante Optuna para la métrica de coherencia, permitió pasar de valores de -11.36, -5.95 y -0.19, para coherencia UMass, UCI y NPMI respectivamente, a los valores optimizados de -5.46, -1.21 y 0.21. Estos resultados se acercan a los resultados obtenidos de la evaluación del modelo base Latent Dirichlet Allocation (LDA).

Cabe resaltar que para el modelo base Non-negative Matrix Factorization (NFM) se obtuvieron mejores resultados de coherencia -2.92, 0.34 y 0.65; esto se podría explicar debido a que el modelo NMF tiene un mejor comportamiento en textos cortos como lo demostrado en los trabajos (Yan, Guo, Liu, Cheng, & Wang, 2013) (Shi, Kang, Choo, & Reddy, 2018), mientras que BERTopic, al ser un modelo que aprende de la semántica y el contexto, requiere de textos más largos para obtener mejores resultados. En la base de tweets recopilada, el 50% de los textos contienen 20 palabras o menos.

Con respecto a la métrica de diversidad, se obtuvo una mejora del 55% como resultado del proceso de optimización, pasando de 0.43 a un valor optimizado de 0.67, obteniendo resultados comparables con los modelos base LDA (diversidad de 0.70) y NFM (diversidad de 0.65).

En relación a la calidad de los tópicos, se observa en la *Figura 17* que para la mayoría de los tópicos las palabras principales que los definen hacen sentido y se relacionan entre ellas.

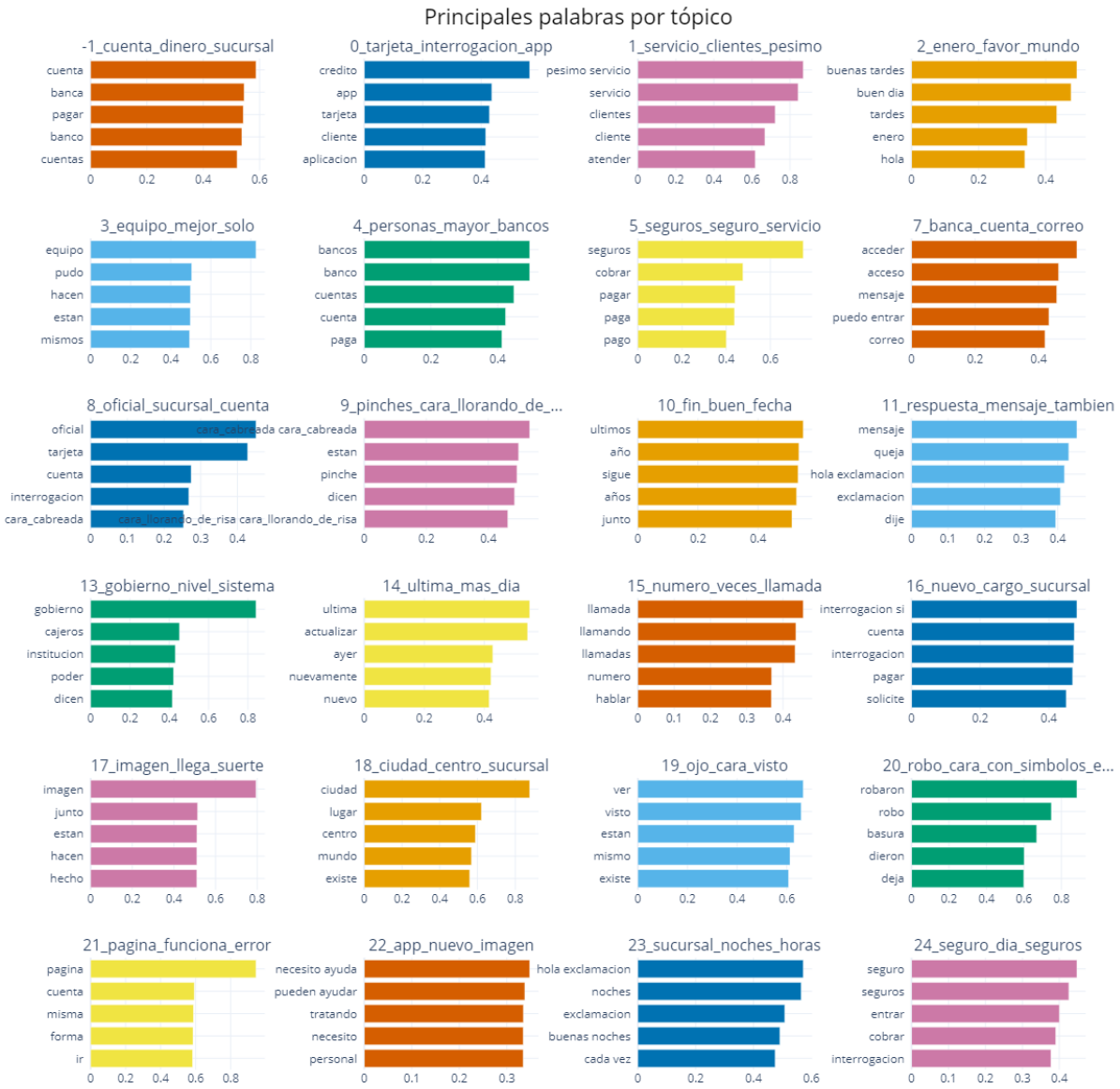


Figura 17. Principales palabras por tópico (exceptuando tópicos 6 y 12).

Fuente: Elaboración propia.

Tópicos como los mostrados en la Figura 18 resultan útiles y son consecuentes con lo observado durante la inspección manual de los comentarios. Por ejemplo, el tópico

0_tarjeta_app_crédito, hace referencia principalmente a problemas con la aplicación móvil del banco, con comentarios como:

- *"@BAztecaAyuda Hola, en la app ya no me aparece activo el botón de pagar el mínimo u otra cantidad de mi tarjetade crédito, solo aparece el de pago total. ¿Qué puedo hacer? Gracias."*
- *"@BBVARe_mx en la app me indica que el servicio está temporalmente inactivo , porque?"*

El tópico *3_personas_mama_mayor*, contempla problemas de interacciones de personas mayores con el banco:

- *"Oyeeee @Banorte_mx que mal servicio tienen para con los adultos mayores,el día de hoy mi mamá quiso retirar su pensión del bienestar en un cajero de la sucursal Insurgentes Tabacalera y no le dio nada, le retuvo su dinero, pero bien abusados ustedes le cobraron 2 comisiones!!!!"*
- *"@BAztecaAyuda quisiera que me ayudaran con un problema que tengo desde el 2019 ya que no he podido sacar el dinero de la pensión de mi mamá que es de adulto mayor iess pensión del bienestar ahora con el cambio de tarjetas me van a dar una nueva y tengo que sacar el dinero, Cómo?"*

El tópico *4_seguros_seguro_servicio*, agrupa comentarios asociados a los seguros vendimos por los bancos:

- *"Oye @ScotiabankMX por qué vendes seguros fraudulentos de @MAPFRE_MX me están cobrando un seguro que no contraté"*
- *"Si tienen la oportunidad de contratar un seguro de autos con Seguros @Banorte_mx, desaprovéchenla"*

Y tópico *6_banca_cuenta_correo*, abarca comentarios de problemas de contraseña y acceso a la banca en línea:

- *"@Citibanamex @ContactoCitibmx tiene días que no puedo entrar a la banca y tampoco me deja ingresar poniendo la contraseña. Pueden apoyarme? Me urge entrar"*
- *"De esos días que Prevencion Fraudes @Citibanamex te llama para cambiar tu contraseña de la banca en línea por una genérica que en 24hrs podrás cambiar y personalizar! En serio?"*

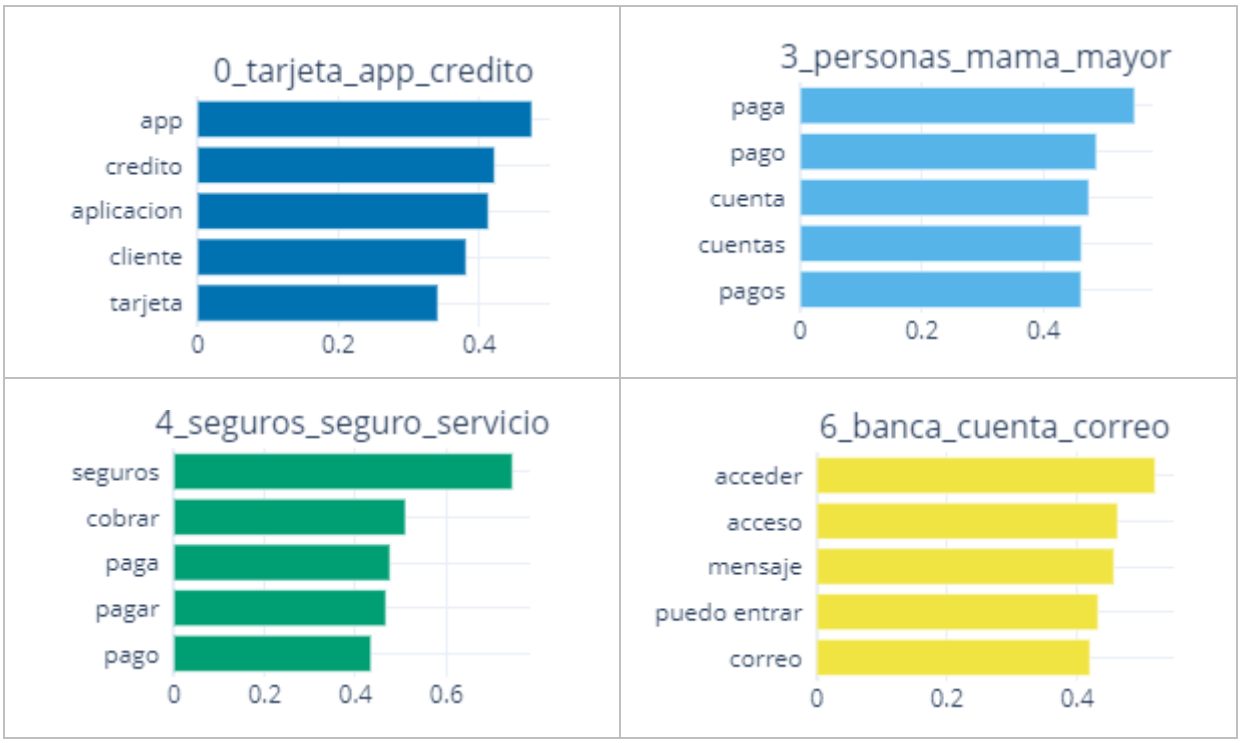


Figura 18. Tópicos interesantes encontrados.

Fuente: Elaboración propia.

Llaman la atención los tópicos 6 y 12 (Figura 19), en los cuales se agruparon los comentarios con emojis o con signo de interrogación. Estos tópicos no hacen mucho sentido para la interpretación de los resultados y para fines prácticos no son útiles. Pudiera haber sido conveniente no aplicar estos pasos de preparación de texto (capítulo 2.3); sin embargo, cabe recordar que durante el proceso de optimización se consideró como hiperparámetros a optimizar la utilización de los métodos para reemplazo de emojis y signos de interrogación, y el resultado de la optimización encontró que el uso de ambos métodos daba mejores resultados para las métricas de coherencia y diversidad.

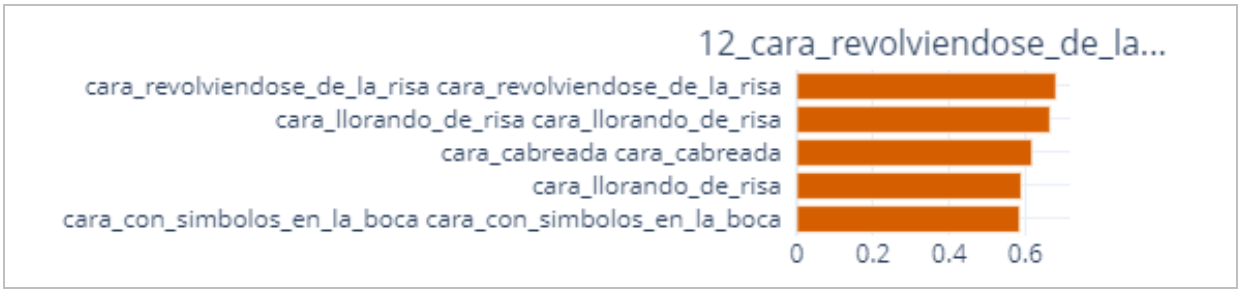


Figura 19. Tópico 6 y Tópico 12.

Fuente: Elaboración propia.

Como lo señalan (Hoyle, y otros, 2021) la evaluación automática de tópicos, en especial a través de la medición de la coherencia tiene grandes áreas de oportunidad, en parte debido a la generalizaciones realizadas en el cálculo de la coherencia para adaptarse a la mayor cantidad posible de escenarios. Por otra parte debido a que las métricas de coherencia fueron desarrolladas en su momento para evaluar modelos que hoy en día se consideran “viejos”, y que no funcionan de la mejor manera con metodologías más recientes como lo son modelos de tópicos neuronales.

Conclusiones

La medición de la satisfacción del cliente en el sector bancario es esencial para mejorar la calidad del servicio. Las encuestas tradicionales tienen limitaciones en cuanto a costo, tiempo, granularidad de los resultados y representatividad estadística. En este contexto, el modelado de tópicos a partir de textos cortos obtenidos de redes sociales resulta prometedor como una alternativa para complementar las deficiencias de los métodos tradicionales.

El empleo de herramientas de modelado de tópicos, como BERTopic, resulta conveniente al integrar en un único modelo todos los pasos necesarios en el modelado de tópicos. Sin embargo, la optimización de los hiperparámetros es un proceso complejo que requiere el uso de técnicas como optimización Bayesiana para lograr dicho objetivo.

Con respecto al proceso de optimización, la propuesta de este trabajo fue realizarlo en dos etapas. En la primera etapa, se seleccionaron los algoritmos de embedding, reducción dimensional y agrupamiento. En la segunda etapa, se optimizó la configuración de la limpieza de texto y los hiperparámetros de los modelos seleccionados. De acuerdo con lo observado, este enfoque permitió reducir los tiempos de entrenamiento a la vez que se observó una clara convergencia hacia valores de coherencia y diversidad estables una vez que el proceso hubo iterado el suficiente número de ciclos.

La librería Optuna facilitó considerablemente la automatización del proceso de optimización, al simplificar la definición del espacio de búsqueda. Su eficiencia en la búsqueda y la capacidad de ejecutar múltiples hilos de optimización en paralelo redujeron considerablemente el total del proceso.

El proceso de optimización dio como resultado que la combinación del modelo de embedding *paraphrase-multilingual-MiniLM-L12-v2*, el algoritmo UMAP para la

reducción dimensional y el algoritmo HDBSCAN para el agrupamiento, dieron los mejores resultados en términos de las métricas de desempeño evaluadas.

Los resultados obtenidos en este trabajo muestran una mejora significativa de los valores de coherencia y diversidad al evaluar el modelo BERTopic inicial en comparación con el modelo BERTopic optimizado. La optimización en 2 etapas permitió mejorar el modelo inicial acercándose a los resultados de los modelos base Latent Dirichlet Allocation y Non-negative Matrix Factorization.

En general, la calidad de los tópicos generados fue buena, con palabras principales que se relacionaban entre sí y tenían sentido en el contexto. Sin embargo, como lo señalan algunos estudios recientes, la generación y optimización automática de modelos de tópicos tiene áreas de oportunidad, al estar basadas principalmente en el uso de la coherencia la cual no es fiable para todos los casos de uso.

Bibliografía

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, (págs. 2623-2631).

Azevedo, A., & Santos, M. (2008). KDD, SEMMA and CRISP-DM: a parallel overview. *ISCAP - Sistemas de Informação - Comunicações em eventos científicos*. Obtenido de http://hdl.handle.net/10400.22/136

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python.* O'Reilly Media Inc.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research 3*, 993-1022.

Bouma, G. J. (2009). Normalized (pointwise) mutual information in collocation extraction. *Proceedings of the Biennial GSCL Conference 2009*, (págs. 31-40).

Churchill, R., & Singh, L. (12 de enero de 2022). The Evolution of Topic Modeling. *ACM Computing Surveys*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1* (págs. 4171-4186). Minneapolis, Minnesota: Association for Computational Linguistics.

Dieng, A. B., Ruiz, F. J., & Blei, D. M. (2020). Topic Modeling in Embedding Spaces. *Transactions of the Association for Computational Linguistics, 8*, 439-453.

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. En *Automated machine learning: Methods, systems, challenges* (págs. 3-33). Springer International Publishing.

Grootendorst, M. (05 de octubre de 2020). *Creating a class-based TF-IDF with Scikit-Learn*. Recuperado el 19 de enero de 2024, de Maarten Grootendorst: https://www.maartengrootendorst.com/blog/ctfidf/

Grootendorst, M. P. (2022). BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794*.

Gudivada, V., Apon, A., & Ding, J. (2017). Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software, 10*, 1-20.

Hansen, P. (1987). The truncatedSVD as a method for regularization. *BIT Numerical Mathematics*, 534–553.

Harrando, I., Lisena, P., & Troncy, R. (2021). Apples to Apples: A Systematic Evaluation of Topic Models. *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)* (págs. 483-493). INCOMA Ltd.

Hoyle, A., Goel, P., Peskov, D., Hian-Cheong, A., Boyd-Graber, J., & Resnik, P. (2021). Is Automated Topic Model Evaluation Broken?: The Incoherence of Coherence. *Neural Information Processing Systems*.

Hutter, F., Hoos, H., & Leyton-Brown, K. (2014). An Efficient Approach for Assessing Hyperparameter Importance. *Proceedings of the 31st International Conference on Machine Learning, volumen 32* (págs. 754-762). Bejing, China: PMLR.

IBM Corporation. (2024). *¿Qué es la minería de texto?* Recuperado el 29 de junio de 2024, de IBM: https://www.ibm.com/es-es/topics/text-mining

IBM Corporation. (17 de 02 de 2024). *IBM Documentation Help*. Obtenido de Conceptos básicos de ayuda de CRISP-DM: https://www.ibm.com/docs/es/spss-modeler/saas?topic=dm-crisp-help-overview

International Organisation for Standardisation. (2015). *Quality management systems - Requirements (ISO 9001:2015)*. Obtenido de https://www.iso.org/obp/ui/#iso:std:iso:9001:ed-5:v1:en

Kim, T., & Wurster, K. (18 de enero de 2024). *emoji 20.10.0*. Obtenido de PyPI: https://pypi.org/project/emoji/

Liam, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Hardt, M., Recht, B., & Talwalkar, A. (2018). Massively Parallel Hyperparameter Tuning. *arXiv: Learning*. Obtenido de http://arxiv.org/abs/1810.05934

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory, 28*(2), 129-137.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1*, 281-297.

McInnes, L., Healy, J., & Astels, S. (Mar de 2017). hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software, 2*. doi:10.21105/joss.00205

McInnes, L., Healy, J., & Melville, J. (2020). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.

Méndez, G. (2021). *La banca mexicana en números al tercer trimestre del 2021.* Ciudad de México: Deloitte.

Mimno, D., Wallach, H. M., Talley, E., Leenders, M., & McCallum, A. (2011). Optimizing semantic coherence in topic models. *Proceedings of the 2011 conference on empirical methods in natural language processing*, (págs. 262-272).

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *ArXiv*.

Parasuramen, A., Zeithaml, V., & Berry, L. (1988). SERVQUAL: A multiple-item scale for measuring consumer perceptions of service quality. *Journal of Retailing*, 64(1), 12–40.

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 559-572.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830.

Python Software Foundation. (2024). *El tutorial de python*. Recuperado el 29 de junio de 2024, de Python documentation: https://docs.python.org/es/3/tutorial/

Python Software Foundation. (2024). *Python Documentation*. Recuperado el 28 de enero de 2024, de unicodedata — Base de datos Unicode: https://docs.python.org/es/3/library/unicodedata.html

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding. Obtenido de https://openai.com/index/language-unsupervised

Rahimi, H., Mimno, D., Hoover, J., Naacke, H., Constantin, C., & Amann, B. (Marzo de 2024). Contextualized Topic Coherence Metrics. *Findings of the Association for Computational Linguistics: EACL 2024*, págs. 1760–1773.

Reichheld, F. (16 de julio de 2015). *Harvard Business Review*. Obtenido de The One Number You Need to Grow: https://hbr.org/2003/12/the-one-number-you-need-to-grow

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics. Obtenido de http://arxiv.org/abs/1908.10084

Röder, M., Both, A., & Hinneburg, A. (2015). Exploring the Space of Topic Coherence Measures. *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining* (págs. 399–408). New York, NY, USA: Association for Computing Machinery. doi:https://doi.org/10.1145/2684822.2685324

SAS Institute Inc. (11 de febrero de 2024). *SAS® Enterprise Miner™ 14.3: Reference Help*. Obtenido de Introduction to SEMMA: https://documentation.sas.com/doc/en/emref/14.3/n061bzurmej4j3n1jnj8bbjjm1a2.htm

Scikit-learn developers. (16 de Febrero de 2024). *Sklearn.feature_extraction.text.CountVectorizer*. Obtenido de scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Scikit-learn developers. (16 de Febrero de 2024). *Sklearn.feature_extraction.text.TfidFVectorizer*. Obtenido de scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Shi, T., Kang, K., Choo, J., & Reddy, C. K. (2018). Short-Text Topic Modeling via Non-negative Matrix Factorization Enriched with Local Word-Context Correlations. *Proceedings of the 2018 World Wide Web Conference* (págs. 1105–1114). International World Wide Web Conferences Steering Committee.

Terragni, S., Fersini, E., Galuzzi, B., Tropeano, P., & Candelieri, A. (Abril de 2021). OCTIS: Comparing and Optimizing Topic models is Simple! *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 263–270.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberl, M., Reddy, T., Cournapeau, D., . . . SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific. *Nature Methods, 17*, 261-272. doi:10.1038/s41592-019-0686-2

Whistler, K. (Ed.). (12 de agosto de 2023). Obtenido de Unicode Normalization Forms: https://unicode.org/reports/tr15/

Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, 29-39.

X Corp. (27 de enero de 2024). *X Developer Platform*. Obtenido de GET /2/users/:id/mentions: https://developer.twitter.com/en/docs/twitter-api/tweets/timelines/api-reference/get-users-id-mentions

Yan, X., Guo, J., Liu, S., Cheng, X., & Wang, Y. (2013). Learning Topics in Short Texts by Non-negative Matrix Factorization on Term Correlation Matrix. *Proceedings of the 2013 SIAM International Conference on Data Mining (SDM)*, (págs. 749-757).

Zhao, H., Phung, D., Huynh, V., Jin, Y., Du, L., & Buntine, W. (2021). Topic Modelling Meets Deep Neural Networks: A Survey. *arXiv:2103.00498*.

ANEXOS

Anexo A. Parámetros clase PreprocesamientoTexto.

```
class PreprocesamientoTexto():
    def __init__(self, n_hilos= 2, normalizar= True, emojis= True, acentos= True, dinero= True,
caracteresEspeciales= True, repeticionCaracteres= True, stopWords= False, include_cnts= True,
min_length= 5)
```

Anexo B. Pipeline para modelado de tópicos.

```
pipe_topicos = Pipeline([
 ('preprocesamiento', pt.PreprocesamientoTexto(include_cnts        = False,
                                               normalizar          = True,
                                               emojis              = True,
                                               acentos             = True,
                                               dinero              = True,
                                               caracteresEspeciales = True,
                                               repeticionCaracteres = True,
                                               stopWords           = True)),
 ('bertopic', BERTopic(
       embedding_model     = embedding_model,        # Paso 1 - Extraer embeddings
       umap_model          = umap_model,             # Paso 2 - Reducción dimesional
       hdbscan_model       = cluster_model,          # Step 3 - Agrupamiento de Embeddings
       vectorizer_model    = vectorizer_model,       # Step 4 - Tokenización de tópicos
       ctfidf_model        = ctfidf_model,           # Step 5 - Ponderado de pesos
       representation_model = representation_model,  # Step 6 - Representación de tópicos
       language            = language,               # Idioma
       calculate_probabilities = True,
       nr_topics           = "auto"
  ))
])
```

Anexo C. Cálculo de la métrica coherencia.

```
def coherencia(corpus, pipe_topicos, topicos):


    # Preprocesar texto con parámetros originales del modelo
    prepro = pipe_topicos['preprocesamiento'].transform(corpus)
```

```
    pretxt = pipe_topicos['bertopic']._preprocess_text(prepro)

    # Obtener modelo vectorizer (reducción dimensional) del modelo de tópicos y construir tokens
    vectorizer = pipe_topicos['bertopic'].vectorizer_model
    analyzer = vectorizer.build_analyzer()
    tokens = [analyzer(doc) for doc in pretxt]

    # Palabras de tópicos
    topic_words = [[words for words, _ in pipe_topicos['bertopic'].get_topic(topic)] for topic in
range(len(set(topicos))-1)]

    # Cálculo de la métrica
    coherence = TopicCoherence(texts=tokens, measure = 'u_mass')
    return coherence.score({"topics" : topic_words})
```

Anexo D. Cálculo de la métrica diversidad.

```
def diversidad(modelo_topicos, topicos, topk=10):
    # Palabras por tópico
    topic_words = [[words for words, _ in modelo_topicos.get_topic(topico)] for topico in
range(len(set(topicos))-1)]

    # Cálculo de la métrica
    topic_diversity = TopicDiversity(topk=topk)
    return topic_diversity.score({"topics" : topic_words})
```