Análisis de Logs de las APIs de TABi Connect usando Machine Learning: Optimización del proceso de monitoreo de APIs de TABi connect en el proceso automático de cotización de cargas usando Inteligencia Articial.

Jhon Alexander Bedoya Carvajal

Informe de práctica para optar al título de Ingeniero Industrial

Modalidad de Práctica

Semestre de Industria

Asesor

Emerson Giraldo, Magíster (MSc) en Dirección de Operaciones y Logística

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería Industrial

Medellín, Antioquia, Colombia

2025

Cita	(Bedoya Carvajal, 2025)
Referencia **Estilo APA 7 (2020)**	Bedoya Carvajal, J. A. (2025). *Análisis de Logs de las APIs de TABi Connect usando Machine Learning* [Informe de práctica]. Universidad de Antioquia, Medellín, Colombia.

Tabla de contenido

Lista de tablas

Lista de figuras

Siglas, acrónimos y abreviaturas

API	Application Programming Interface
CRISP-DM	Cross Industry Standard Process for Data Mining
EDA	Exploratory Data Analysis
IA	Inteligencia Artificial
LDA	Latent Dirichlet Allocation
ML	Machine Learning
PCA	Principal Component Analysis
RPA	Robotic Process Automation
TF-IDF	Term Frequency - Inverse Document Frequency
BoW	Bag of Words

Resumen

Este proyecto hecho en TABi Connect tiene como objetivo desarrollar un sistema de análisis de logs de APIs utilizando Inteligencia Artificial con el fin de optimizar el monitoreo y la identificación de problemas en su proceso de cotización automática de cargas. El proyecto siguió un enfoque cuantitativo, aplicando la metodología CRISP-DM en cuatro etapas: comprensión del problema, preparación de datos, modelamiento y evaluación de resultados. Se implementó un pipeline de datos que extrae, transforma, carga y limpia los logs desde Amazon CloudWatch utilizando Python. Posteriormente, se ajustaron y evaluaron modelos de Machine Learning no supervisados, como DBSCAN y Latent Dirichlet Allocation (LDA), para identificar patrones y categorizar errores. Los resultados muestran que el modelo LDA clasificó los errores en cuatro temas principales: validación de datos, excepciones internas, errores de servidor y solicitudes externas, con un impacto significativo en la priorización de problemas. Por otro lado, aunque el modelo DBSCAN mostró un buen rendimiento en las métricas Silhouette y Calinski-Harabasz, sus patrones no fueron completamente interpretables desde el punto de vista del negocio. Las implicaciones prácticas incluyen la reducción del tiempo y recursos humanos requeridos para las tareas de monitoreo de las APIs. Además, la jerarquización de errores permite abordar problemas críticos en etapas iniciales, reduciendo así su propagación. Finalmente, se destaca la necesidad de seguir explorando técnicas avanzadas de Machine Learning y Deep Learning para manejar el creciente volumen de datos y mejorar aún más la eficiencia del sistema con patrones más robustos.

Palabras clave: Análisis de Logs, Análisis de Datos, Inteligencia Artificial, Aprendizaje Automático No Supervisado, CRISP-DM, Agrupamiento, Modelado de Temas, DBSCAN, Latent Dirichlet Allocation, LDA.

Abstract

This project, developed in TABi Connect, aims to create a log analysis system for APIs using Artificial Intelligence to optimize monitoring and problem identification in the automated load quoting process. The project followed a quantitative approach, applying the CRISP-DM methodology in four stages: problem understanding, data preparation, modeling, and result evaluation. A data pipeline was implemented to extract, transform, load and clean logs from Amazon CloudWatch using Python. Subsequently, unsupervised Machine Learning models, such as DBSCAN and Latent Dirichlet Allocation (LDA), were tuned and evaluated to identify patterns and categorize errors. The results show that the LDA model classified errors into four main categories: data validation, internal exceptions, server errors, and external requests, significantly impacting problem prioritization. Conversely, while the DBSCAN model performed well on silhouette and Calinski-Harabasz metrics, its patterns were not fully interpretable from a business perspective. The practical implications include reducing the time and human resources required for API monitoring tasks. Additionally, error prioritization enables addressing critical issues at early stages, thereby minimizing their propagation. Finally, the need to continue exploring advanced Machine Learning and Deep Learning techniques is highlighted to handle the growing data volume and further enhance system efficiency with more robust patterns.

*Keywords***:** Log Analysis, Data Analysis, Artificial Intelligence, Unsupervised Machine Learning, CRISP-DM, Clustering, Topic Modeling, DBSCAN, Latent Dirichlet Allocation, LDA.

1. Introducción

TABi Connect es una plataforma innovadora desarrollada por Hubtek que simplifica y optimiza las operaciones logísticas al automatizar tareas repetitivas y de gran volumen. Esta solución permite a los proveedores de servicios logísticos capturar y analizar el 100% de las solicitudes de tarifas (cotizaciones) que fluyen a través de sus procesos, asegurando mayor eficiencia operativa y reduciendo la carga manual en actividades clave. En el núcleo de su funcionamiento se encuentran los logs, que según (Ramachandran, y otros, 2023), son archivos generados por sistemas de seguridad, servidores y aplicaciones, que documentan eventos operacionales y ofrecen un recurso invaluable para entender el comportamiento del sistema y diagnosticar problemas.

Sin embargo, el análisis de estos logs presenta desafíos significativos. Su volumen y complejidad aumentan rápidamente, lo que hace que la inspección manual sea cada vez más difícil, costosa y poco efectiva. Actualmente, los logs de las APIs de TABi Connect son monitoreados mediante reglas de negocio establecidas que resultan tediosas de mantener y no permiten capturar de manera eficiente todas las variaciones de problemas o anomalías. Dado que las cotizaciones automáticas pueden realizarse a través de APIs o Bots de RPA (Robotic Process Automation), una falla en las APIs repercute directamente en la capacidad de cotizar cargas para los clientes, afectando tanto la calidad del servicio como la reputación de la empresa.

Este proyecto tiene como objetivo principal desarrollar un sistema basado en Inteligencia Artificial (IA) y Machine Learning (ML) No Supervisado para analizar los logs de las APIs. Esta solución permitirá optimizar el monitoreo e identificación de problemas. Además de reducir tiempos de inactividad y minimizar costos, el sistema proporcionará herramientas para priorizar fallas, permitiendo al personal técnico dedicar tiempo a actividades estratégicas que agreguen más valor a la organización.

La metodología empleada para el desarrollo del proyecto es CRISP-DM (Cross-Industry Standard Process for Data Mining), un enfoque estructurado que asegura la comprensión profunda del problema de negocio, la preparación de los datos, el ajuste de los modelos y la evaluación de su rendimiento. No obstante, el trabajo enfrenta ciertas limitaciones. El alto volumen y diversidad de datos en los logs representa un desafío técnico considerable para el almacenamiento, procesamiento y análisis. Además, algunos patrones identificados en los logs no siempre son

intuitivos ni están alineados con el contexto del negocio, lo que requiere ajustes adicionales en los modelos. A pesar de estas barreras, el impacto potencial de la solución es significativo, ya que podría transformar la forma en que se monitorean las APIs en la empresa.

En la sección de objetivos se establecen el propósito general y las metas específicas del proyecto. El marco teórico describe los conceptos fundamentales relacionados con el análisis de logs, las técnicas de IA y las herramientas de Machine Learning aplicadas. La metodología detalla las etapas del proyecto bajo el modelo CRISP-DM. Posteriormente, en la sección de análisis de resultados, se presentan la implementación y evaluación de los modelos desarrollados. Finalmente, las conclusiones resumen los hallazgos clave, las limitaciones identificadas y las recomendaciones para la continuación de este trabajo.

2. Objetivos

2.1 Objetivo general

Desarrollar un sistema para el análisis de logs de las APIs de TABI Connect usando Inteligencia Artificial, con el fin de optimizar el proceso de monitoreo e identificación de problemas durante el proceso automático de cotización de cargas.

2.2 Objetivos específicos

- Comprender la importancia de los logs de las APIs de TABi Connect y cómo un análisis automatizado y una rápida identificación de problemas puede beneficiar a la empresa.
- Implementar un pipeline de datos que extraiga, transforme y cargue los logs a un entorno de análisis adecuado desde Amazon CloudWatch.
- Ajustar un modelo No Supervisado de Machine Learning para la identificación de problemas en los logs de las APIs de TABi Connect.
- Analizar y evaluar los resultados obtenidos del modelo aplicando conocimiento del negocio.

3. Marco teórico

(Koçi, Franch, Jovanovic, & Abelló, 2020) Explican que las API (Application Programming Interfaces) son conjuntos de reglas y protocolos que permiten la comunicación entre diferentes aplicaciones de software y que pueden usarse para varios propósitos, como recuperar datos, enviar datos o realizar operaciones en un servidor. Desarrollaron un enfoque basado en datos para medir la usabilidad de las APIs web, analizando registros de uso, preprocesando los datos (fusión, limpieza, estructuración y generalización) y se definiendo métricas como claridad, consistencia y tasas de error repetido. Estas métricas sirvieron para entrenar un modelo de clasificación basado en árboles de decisión, que alcanzó una precisión del 72.25% al identificar endpoints con problemas de usabilidad. Por otro lado, (Ramachandran, y otros, 2023) describen los logs como archivos creados por sistemas de seguridad, dispositivos de red, servidores y diversas aplicaciones de software como APIs que rastrean los eventos operacionales de un programa. Estos registros sirven como recursos valiosos para extraer información útil sobre el comportamiento del sistema. La complejidad y la cantidad de datos contenidos en los archivos de registro están aumentando con rapidez y su análisis es efectivo para encontrar la causa raíz de problemas. Según (He P. , Zhu, He, Li, & Lyu, 2016), log parsing es el proceso de convertir logs no estructurados en eventos estructurados, permitiendo así un análisis automatizado y eficiente de estos. Este proceso identifica y separa las partes constantes y variables de los logs. Las partes constantes corresponden a plantillas o eventos predefinidos, mientras que las variables contienen información específica del sistema en ejecución, como direcciones IP, identificadores de bloques o marcas temporales.

El modelo CRISP-DM (Cross Industry Standard Process for Data Mining) es descrito por (Massahiro Shimaoka, Cordeiro Ferreira, & Goldman, 2024) como un estándar ampliamente adoptado en proyectos de ciencia de datos, gracias a su estructura flexible y adaptable. Consta de seis fases: comprensión del negocio, comprensión de los datos, preparación de datos, modelado, evaluación y despliegue, que aseguran un enfoque sistemático y orientado a los objetivos del proyecto.

La inteligencia artificial (IA) es un conjunto de tecnologías que permiten que las computadoras realicen una variedad de funciones avanzadas, incluida la capacidad de ver, comprender y traducir lenguaje hablado y escrito, analizar datos, hacer recomendaciones y mucho más (Google LLC, s.f.). Por otro lado, (Sarker, 2021) menciona en su trabajo que Machine

Learning (ML) es una rama de la Inteligencia Artificial (IA) que se centra en el desarrollo de algoritmos y modelos estadísticos que permiten a las computadoras realizar tareas específicas sin ser programadas explícitamente para ello. En lugar de seguir instrucciones predefinidas, los sistemas de Machine Learning aprenden de los datos, identificando patrones y tomando decisiones basadas en esa información. En (Sarker, 2021) se presenta una revisión integral de los algoritmos de aprendizaje automático, incluyendo técnicas supervisadas, no supervisadas, semisupervisadas y por refuerzo, junto con su aplicabilidad en casos reales y direcciones futuras de investigación; explorando aplicaciones en ciberseguridad, ciudades inteligentes, salud, comercio electrónico y análisis de sentimientos, destacando los principios y capacidades de cada técnica. El estudio dio como resultado una visión clara de cómo estas técnicas pueden mejorar la inteligencia de aplicaciones reales. Según (Sarker, 2021), el Machin e Learning abarca varios tipos de enfoques de aprendizaje, incluidos el supervisado, el no supervisado, el semi-supervisado y por refuerzo, cada uno adecuado para diferentes tipos de problemas y características de los datos. Además, explica que en el Machine Learning No supervisado el modelo se entrena con datos que no están etiquetados, donde el objetivo es identificar patrones o estructuras subyacentes en los datos. De manera similar, (Holdsworth & Scapicchio, 2024) destaca en su trabajo que el Deep Learning (DL) es un subconjunto del Machine Learning (ML) que utiliza redes neuronales multicapa, llamadas redes neuronales profundas, para simular el complejo poder de toma de decisiones del cerebro humano y que la principal diferencia entre el Deep Learning (DL) y el Machine Learning (ML) es la estructura de la arquitectura de red neuronal subyacente. Finalmente, (Just, 2024) describe el Natural Language Processing (NLP) como una subdisciplina de la Inteligencia Artificial (IA) que combina la lingüística computacional con modelos estadísticos, de aprendizaje automático y de aprendizaje profundo para permitir que las máquinas comprendan, interpreten y generen lenguaje humano de manera que sea útil para diversas aplicaciones.

Amazon CloudWatch es un servicio de monitorización de AWS de recursos de la nube y de las aplicaciones que se ejecutan en AWS (Amazon AWS, s.f.). Mientras que (Ranjan, y otros, 2023) mencionan en su investigación de las aplicaciones de Python en ciencia de datos que este es un lenguaje de programación avanzado y versátil ampliamente utilizado en ciencia de datos por su facilidad de uso, estabilidad y una extensa biblioteca de herramientas especializadas en análisis, modelado y visualización de datos. (Ranjan, y otros, 2023) se centraron en analizar cómo Python y sus entornos integrados de desarrollo pueden potenciar aplicaciones en ciencia de datos y

machine learning, realizando una revisión de las bibliotecas más relevantes, como Pandas, NumPy, TensorFlow y Matplotlib, evaluando su capacidad para simplificar procesos como el análisis de datos, la visualización y el desarrollo de modelos de aprendizaje automático y llegando a la conclusión de que Python, gracias a su ecosistema robusto y comunidad activa agiliza los procesos de investigación y permite abordar problemas complejos en diversas áreas. Además, (Peng, y otros, 2021) definen el análisis exploratorio de datos (EDA) como una etapa clave en los proyectos de ciencia de datos que permite comprender los datos mediante la manipulación y visualización, con el objetivo de descubrir patrones, relaciones y problemas como valores faltantes o anomalías. Por otro lado, (Recabal Ávila, 2024) explica que el formato Parquet es un tipo de almacenamiento de datos basado en columnas que optimiza la compresión y la lectura selectiva de datos, permitiendo manejar grandes volúmenes de información de manera eficiente. (Recabal Ávila, 2024) aplicó técnicas de Big Data para procesar datos del sector eléctrico chileno con el objetivo de optimizar rutinas existentes y mejorar tiempos de procesamiento. Se utilizaron formatos como Parquet y Feather, para reducir significativamente los tamaños de archivo y mejorar la velocidad de lectura y escritura. Como resultados, el procesamiento de los datos, que inicialmente tomaba horas, se redujo a minutos, con tamaños de archivo disminuidos de cientos de gigabytes a menos de 20 gigabytes, demostrando mejoras en la eficiencia y escalabilidad del manejo de datos.

El modelo Latent Dirichlet Allocation (LDA), introducido por Blei, Ng y Jordan (2003), es explicado por (Blei, Ng, & Jordan, 2003) como un enfoque probabilístico que fue diseñado para modelar colecciones de datos discretos, como colecciones de textos o documentos. LDA asume que los documentos son combinaciones aleatorias de temas latentes, y cada tema se define como una distribución sobre palabras. Este modelo utiliza un enfoque jerárquico bayesiano de tres niveles: a nivel del corpus se determinan los parámetros generales; a nivel del documento se generan mezclas específicas de temas; y, a nivel de palabra, cada término es asignado a un tema. Esto permite representar documentos mediante distribuciones de temas. Además, (Pinto Gurdiel, Morales Mediano, & Cifuentes Quintero, 2021) explican que, en el campo del modelado de tópicos, dos métricas fundamentales para evaluar la calidad de los modelos y determinar el número óptimo de tópicos son la perplexity y la coherence. La perplexity evalúa el desempeño predictivo del modelo, midiendo qué tan bien generaliza al predecir documentos no observados; una menor perplexity indica mejor generalización, pero puede no reflejar coherencia semántica en los tópicos. Por otro lado, la coherence se centra en la validación semántica, asegurando que los tópicos estén

compuestos por palabras que los humanos consideran relacionadas. (Pinto Gurdiel, Morales Mediano, & Cifuentes Quintero, 2021) exploraron ambas métricas en el contexto del análisis de entrevistas, observándose que la coherence permitió identificar temas más interpretables y relevantes, mientras que la perplexity destacó en simplificar la estructura del modelo. Finalmente, (Sievert & Shirley, 2014) detallan LDAvis como una herramienta interactiva basada en la web que facilita la visualización y comprensión de los modelos de temas generados mediante Latent Dirichlet Allocation (LDA). Esta técnica aborda uno de los mayores desafíos del modelado de temas, que es interpretar y explorar distribuciones complejas de términos y temas en grandes conjuntos de textos.

En su trabajo, (Egger, 2022) explica en su trabajo que el método Bag of Words (BOW) es una representación textual en la que se descompone un documento en sus palabras individuales para cuantificar su frecuencia y permitir análisis como clasificación o comparación de documentos. (Egger, 2022) Evaluó diferentes técnicas de representación de texto, incluyendo BOW, TF-IDF y modelos avanzados como Word2Vec y BERT, con el objetivo de analizar textos del ámbito del turismo y explorar su potencial en aplicaciones como análisis de sentimiento y clasificación de documentos, encontrando que técnicas como BOW y TF-IDF resultaron útiles para tareas menos complejas. Por otro lado, (Zhou, y otros, 2020) describe el uso de TF-IDF como un tokenizador para convertir palabras de un texto en vectores numéricos que representan su relevancia dentro de un conjunto de textos o documentos, combinando la frecuencia de las palabras en un documento y su aparición inversa en otros documentos. En (Zhou, y otros, 2020) se optimizó el algoritmo TF-IDF integrándolo con CountVectorizer para mejorar el procesamiento de textos y preservar el mapeo palabra-índice. Este enfoque, aplicado al clustering de noticias con LDA, mostró una reducción de la perplexity del modelo y mayor velocidad de procesamiento en grandes volúmenes de datos, aunque con un alto consumo de memoria en escalas mayores. En (Wang, y otros, 2022) describen Word2Vec como una técnica de procesamiento de lenguaje natural que convierte palabras en vectores numéricos, preservando sus relaciones semánticas. (Wang, y otros, 2022) desarrollaron LogUAD, un método no supervisado basado en Word2Vec para detectar anomalías logs, extrayendo características de los mensajes originales mediante Word2Vec y TF-IDF que luego se agruparon utilizando K-means para identificar registros anómalos, obteniendo que LogUAD superó a métodos tradicionales como LogCluster, logrando una mejora del 67.25% en la F1-score. Finlamente, (Wahyuningsih & Chen, 2024) definen PCA (Análisis de Componentes

Principales) como una técnica de reducción de dimensionalidad que transforma los datos originales en componentes principales, reteniendo la mayor cantidad posible de información relevante y utilizaron PCA para reducir las dimensiones de características generadas por la vectorización TF-IDF.

En el ámbito del análisis de logs en la seguridad informática, (Riadi, Istiyanto, Ashari, & Subanar, 2013) desarrollaron el módulo NFAT (Network Forensic Analysis Tool) para apoyar la investigación de redes mediante el uso del algoritmo de agrupamiento K-means, clasificando datos en tres categorías de ataques: peligrosos, medianamente peligrosos y no peligrosos. El sistema, diseñado con software de código abierto (Linux, Unix, PHP y MySQL), procesó logs almacenados en una base de datos, agrupándolos por proximidad y etiquetando los grupos según su nivel de peligrosidad y permitió identificar fuentes y objetivos de ataques en redes, facilitando investigaciones forenses y aportando información de valor para procesos judiciales, aunque los resultados dependieron de la inicialización aleatoria de los centroides.

En (Sandhu & Mohammed, 2022) se abordó la detección de anomalías en registros (logs) mediante técnicas de procesamiento de lenguaje natural (NLP), combinadas con métodos como TF-IDF y representaciones de palabras (embeddings). Este trabajo incluyó dos aproximaciones: NLP junto con TF-IDF para entrenar modelos de clasificación (regresión logística y SVM) y embeddings con una red neuronal convolucional (CNN) y una capa LSTM. Los resultados mostraron que la segunda aproximación superó a la primera en precisión, alcanzando un 90% de efectividad en 30 épocas, en comparación con el 76-79% obtenido mediante la técnica TF-IDF. Además, (Bertero, Roy, Sauvanaud, & Tredan, 2017) exploran la aplicación de algoritmos de Natural Language Processing (NLP) para la minería de logs, específicamente para detección de anomalías con una intervención humana mínima. Se utilizaron modelos de aprendizaje automático como Naive Bayes, Random Forest y redes neuronales y las técnicas aplicadas incluyeron la transformación de logs en vectores multidimensionales de características y el uso de algoritmos de NLP para extraer información relevante. Los resultados mostraron que se podían entrenar clasificadores con una precisión de aproximadamente el 90% sin intervención humana.

4. Metodología

El proyecto se desarrolla siguiendo un enfoque cuantitativo, debido a la naturaleza del Análisis de Datos y el uso de Inteligencia Artificial y Modelos de Machine Learning. La metodología cuenta con cuatro etapas clave de la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining): comprensión del problema, entendimiento y preparación de datos, modelamiento y validación del modelo.

En la fase inicial de comprensión del problema se busca entender la relevancia de los logs de las APIs de TABi Connect y cómo el análisis automatizado puede optimizar la identificación temprana de problemas. Las actividades incluyen estudiar el proceso actual de identificación de problemas utilizando logs y acceder a los logs almacenados en Amazon CloudWatch. El resultado de esta etapa es la documentación del proceso y un plan de proyecto claro para guiar las fases subsiguientes.

La segunda etapa, entendimiento y preparación de los datos, se centra en implementar un pipeline de datos que permita extraer, transformar y cargar los logs en un entorno adecuado para el análisis. Para ello, se realiza la conexión a Amazon CloudWatch mediante Python y Boto3, seguida de la extracción de los logs en formato parquet. Posteriormente, los datos son limpiados y transformados, asegurando un formato uniforme. Finalmente, se lleva a cabo un análisis exploratorio de los datos (EDA) para comprender mejor su estructura y contenido. El producto final de esta etapa es una base de datos de logs limpia y lista para el análisis, junto con un informe del EDA.

La siguiente fase es el modelamiento, en esta se ajusta un modelo de Machine Learning no supervisado para identificar patrones en los logs. Las actividades incluyen preprocesar los datos en un formato adecuado para el modelo, ajustar diferentes modelos no supervisados y evaluar la necesidad de realizar ajustes adicionales en la transformación de los datos. Con base en estos análisis, se selecciona el modelo más adecuado. El resultado de esta etapa es un modelo no supervisado que identifica patrones en los logs.

Finalmente, se evalúan y analizan los resultados del modelo ajustado, usando métricas adecuadas y aplicando conocimiento del negocio para determinar su efectividad. Obteniendo como resultado de esta etapa la calidad de los resultados del modelo y la decisión de si el modelo es apto para integrarlo al proceso de monitoreo de las APIs.

5. Análisis de resultados

5.1. Comprensión del proceso y la importancia de los logs y las APIs

5.1.1. Problema

En la actualidad, el monitoreo y análisis de los logs generados por las APIs de TABi Connect se realiza de manera manual, utilizando reglas de negocio que resultan tediosas, ineficientes y difíciles de mantener a medida que aumentan la complejidad y el volumen de datos no estructurados. Este enfoque limita la capacidad de la empresa para identificar, de forma oportuna y precisa, problemas que afectan el funcionamiento de las APIs, lo que puede derivar en interrupciones en los procesos de cotización automática y, consecuentemente, en una disminución en la calidad del servicio prestado a los clientes.

El desafío radica en implementar un sistema basado en Analítica e Inteligencia Artificial (IA), que permita analizar de manera eficiente los logs de las APIs, identificando problemas de forma más rápida para garantizar un buen servicio. Esto es relevante para optimizar recursos, reducir costos y mejorar la experiencia del cliente.

5.1.2. Proceso de monitoreo actual

1. Los eventos y errores generados por las APIs se registran automáticamente en Amazon CloudWatch.
2. Un equipo técnico revisa manualmente los logs para identificar errores relevantes. Dicha revisión sigue reglas y filtros de negocio definidas.
3. El equipo intenta diagnosticar y solucionar los problemas encontrados.
4. Si el problema no puede resolverse internamente, se escala al equipo responsable del componente afectado.

5.1.3. Diseño de solución

1. Los eventos y errores generados por las APIs se registran automáticamente en Amazon CloudWatch.

2. Se configura una extracción y limpieza automática cada hora de los logs con errores desde Amazon CloudWatch mediante la librería Boto3.
3. Se categorizan los logs usando el modelo no supervisado ajustado. Estos resultados se almacenan en un reporte en una ruta compartida con el equipo técnico encargado del monitoreo.
4. El equipo de monitoreo revisa los resultados de la categorización del modelo. Los errores más críticos se priorizan y se destacan en el reporte automático generado.
5. El equipo intenta solucionar los problemas encontrados.
6. Si el problema no puede resolverse internamente, se escala al equipo responsable del componente afectado.

5.2. Implementación de pipeline de datos

5.2.1. Extracción de datos

Los logs generados en Amazon CloudWatch se organizan en Log Groups, que contienen Log Sequences. Para este proyecto, se seleccionaron siete Log Groups relevantes. Utilizando la librería Boto3, se establece una conexión a Amazon CloudWatch para extraer exclusivamente los logs que representan errores (aquellos con códigos de estado mayores o iguales a 400), incluyendo información clave como tipos de logs (logs y responses), dentro del periodo comprendido entre el 01/10/2024 y el 10/11/2024. De cada log, se extrae el campo Message, que contiene toda la información necesaria para el análisis. Estos datos se almacenan inicialmente en un dataframe de Pandas y luego se guardan localmente en formato Parquet, optimizando el espacio de almacenamiento y los tiempos de carga.

5.2.2. Limpieza y procesamiento

Los datos extraídos se procesan en un entorno de Python, comenzando con su conversión a formato JSON, como se muestra en la *Figura 1*, para facilitar el log parsing, es decir, la extracción y estructuración de campos específicos como Type, API, Domain, Status Code, Detail y Summary. Posteriormente, se realiza la limpieza del campo Detail, el cual contiene mensajes en lenguaje natural. Este proceso incluye el uso de expresiones regulares para eliminar caracteres especiales, números y otros elementos no alfabéticos, así como la exclusión de stop words, palabras que no

aportan valor analítico. El resultado se muestra en la *Tabla 1*. Finalmente, los datos estructurados y procesados se almacenan en un nuevo dataframe de Pandas, el cual se guarda en formato Parquet para asegurar una manipulación eficiente en las etapas posteriores del análisis.

Figura 1. Estructura de los logs en JSON


```
{'event': {'track_type': 'RESPONSE',
  'api': 'DB',
  'response': {'uri': 'http://db-api.gotabi.ai/                              ',
   'status_code': 400,
   'headers': {'content-length': '198',
    'content-type': 'application/json',
    'x-t-id': '202411-0900-5959-C4DB06A5-8EB6-4EC1-8A0C-51D4E9EDDE90',
    'x-process-time': '0.0013839940074831247'},
   'content': {'transaction_id': '202411-0900-5959-C4DB06A5-8EB6-4EC1-8A0C-51D4E9EDDE90',
    'detail': [{'type': 'json_invalid',
      'loc': ['body', 38],
      'msg': 'JSON decode error',
      'input': {},
      'ctx': {'error': 'Expecting value'}}]}}},
 'api': 'DB',
 'forwarded_for': '3.18.162.178, 18.68.5.113',
 'forwarded_proto': 'http',
 'transaction_id': '202411-0900-5959-C4DB06A5-8EB6-4EC1-8A0C-51D4E9EDDE90',
 'forwarded_port': '80',
 'amz_cf_id': 'Ga3oRBS99_OCn2B5MYGo1gfWQwUMemhP4SQoro40d8M3Zpxm06HwcA==',
 'x_amzn_trace_id': 'Root=1-672eb40f-004fd709680803ac01b44c71',
 'level': 'info',
 'timestamp': '2024-11-09T00:59:59.956061Z'}
```


Fuente: Elaboración propia

Tabla 1. Estructura de los logs parseados

Llave	**Valor**
type	RESPONSE
api	DB
status_code	400
detail	json decode error expecting value
summary	None
domain	db-api.gotabi.ai

Fuente: Elaboración propia

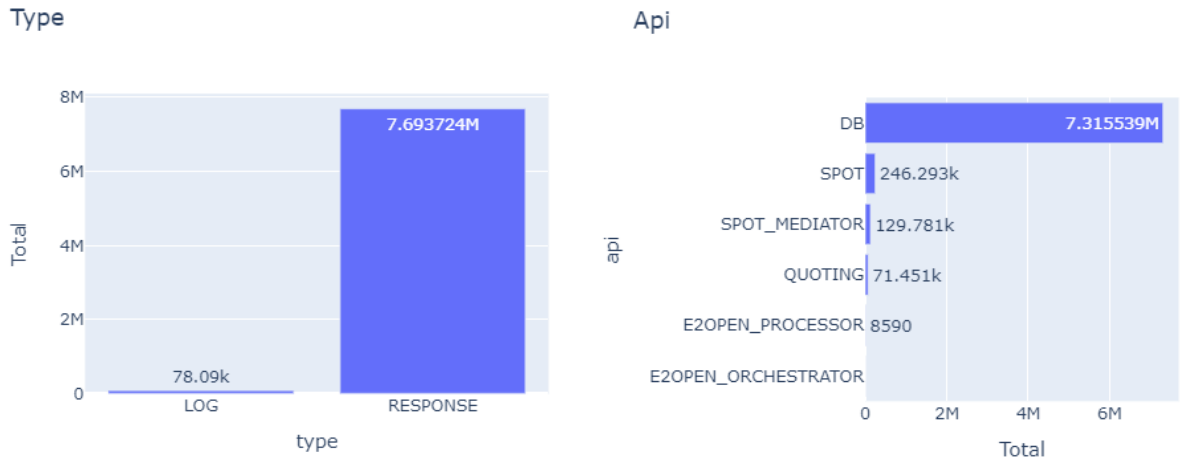
Adicional, una vez estructurados los datos, se crea otra versión de los logs en los que se concatenan los valores y son tratados como textos o documentos. Esto con el fin de poder analizar cuáles son los logs más comunes e identificar representaciones importantes en el corpus. El ejemplo de la *Tabla 1* adopta la siguiente estructura:

response db 400 json decode error expecting value db_api_gotabi_ai

5.2.3. Análisis exploratorio

En este proyecto, el EDA se centra en identificar tendencias y distribuciones en los logs de las APIs de TABi Connect. Esta etapa es crucial para garantizar la calidad de los datos procesados y para identificar las transformaciones adicionales necesarias, de modo que el modelo de Machine Learning pueda recibir insumos óptimos para su entrenamiento. A través de técnicas visuales, se busca identificar posibles inconsistencias o valores atípicos.

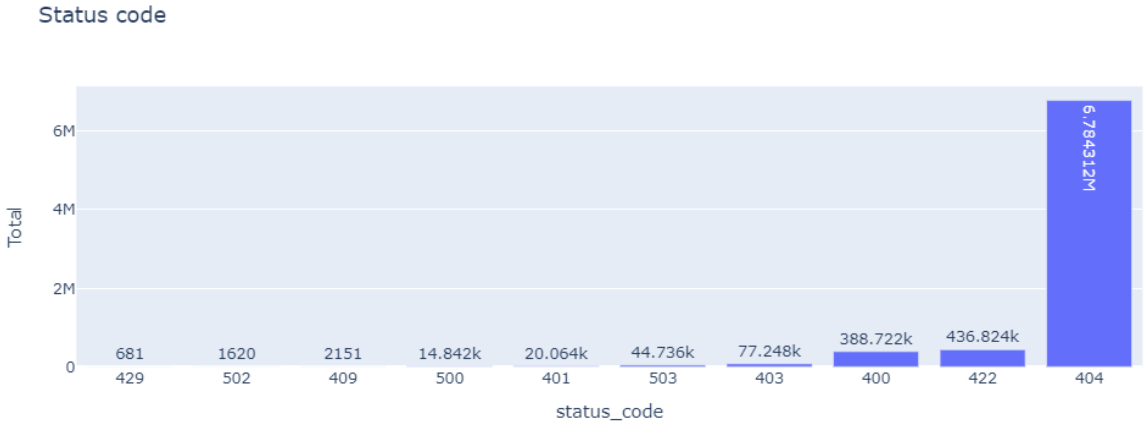
Figura 2. Distribución Tipo y Api



Fuente: Elaboración propia

En la *Figura 2* se evidencia la distribución de logs de acuerdo con su tipo, siendo los tipos “RESPONSE” el tipo de logs dominante entre las dos clases que se eligieron para este análisis. Además, que la api “DB” es la más representativa en los logs con errores.

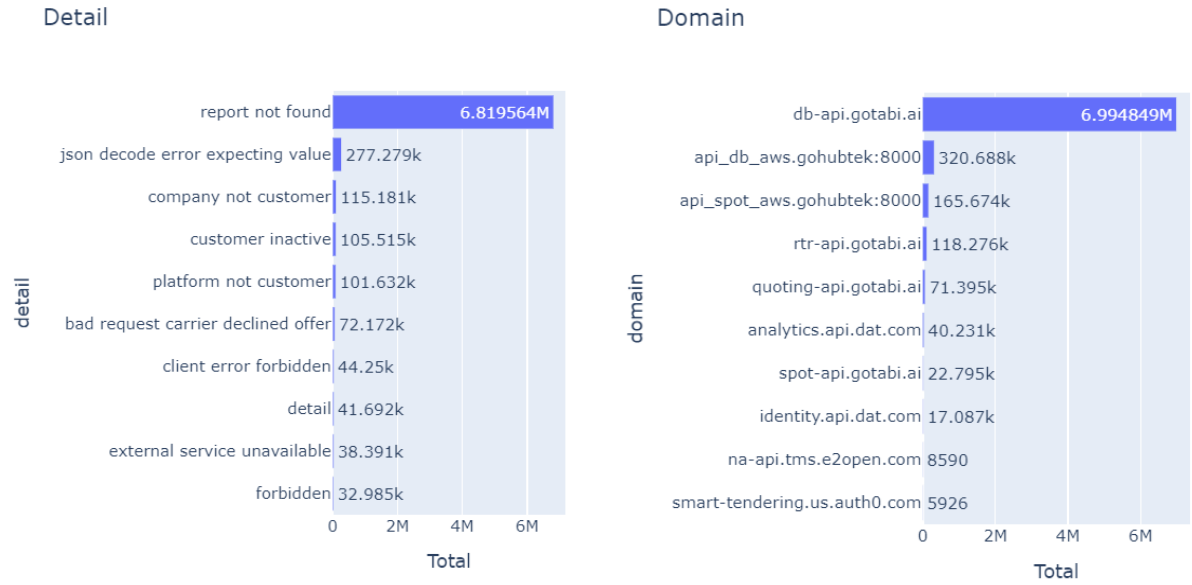
Figura 3. Distribución Status code



Fuente: Elaboración propia

La *Figura 3* muestra que el status code 404 es, por mucho, el más común en los logs que representan errores en la APIs.

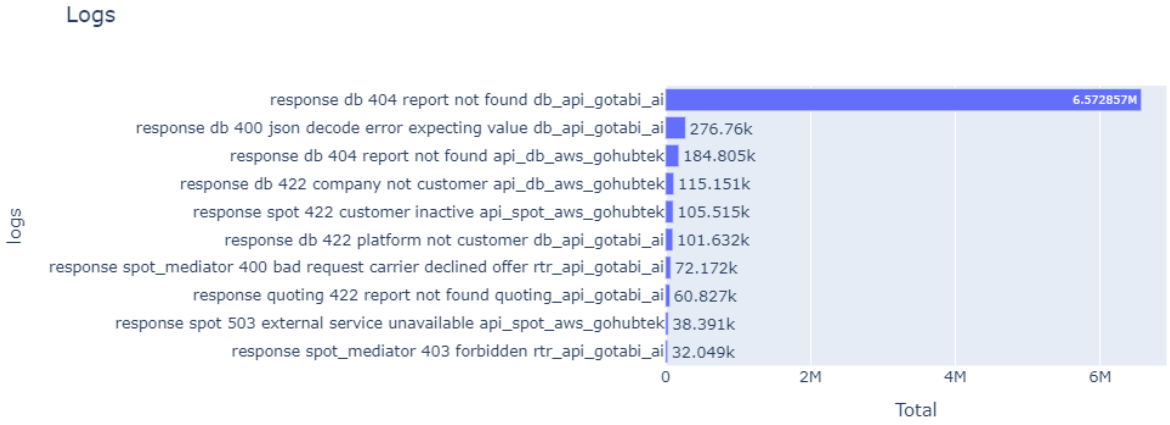
Figura 4. Distribución Detail y Domain



Fuente: Elaboración propia

En la *Figura 4* se observa que “report not found” es un detalle o mensaje de error extremadamente común. De manera similar, el dominio “db-api.gotabi.ai” es muy frecuente en los logs con errores.

Figura 5. Distribución logs documents



Fuente: Elaboración propia

Tabla 2. Frecuencias logs documents

logs	Frecuencia Absoluta	Frecuencia Relativa
response db 404 report not found db_api_gotabi_ai	6572857	84,573%
response db 400 json decode error expecting va...	276760	3,561%
response db 404 report not found api_db_aws_go...	184805	2,378%
response db 422 company not customer api_db_aw…	115151	1,482%
response spot 422 customer inactive api_spot_a...	105515	1,358%
response db 422 platform not customer db_api_g...	101632	1,308%
response spot_mediator 400 bad request carrier...	72172	0,929%
response quoting 422 report not found quoting_...	60827	0,783%
response spot 503 external service unavailable...	38391	0,494%
response spot_mediator 403 forbidden rtr_api_g...	32049	0,412%

Fuente: Elaboración propia

En la *Figura 5* y la *Tabla 2* se analiza la frecuencia de los logs creados a partir de los logs parseados, encontrando que el log “response db 404 report not found db-api.gotabi.ai” está muy presente en los datos. Este log en particular no aporta valor analítico, por lo que podría considerarse dejarlos fuera si la frecuencia de los logs es relevante para el análisis. En este caso, se eliminaron duplicados de los logs. Los registros duplicados no aportan información nueva, generan redundancia y pueden introducir sesgos en métricas y modelos, tal como lo hizo en las visualizaciones al inflar artificialmente la frecuencia de ciertos eventos. Además, mantener duplicados incrementa innecesariamente el tamaño de los datos, afectando la eficiencia de almacenamiento y

procesamiento. Al eliminar duplicados, se optimizan los recursos computacionales, se facilita la interpretación de resultados y se asegura que cada log represente un evento único.

Figura 6. Nube de palabras con Bag of Words

Fuente: Elaboración propia

Bag of Words es una técnica que convierte el texto en una representación numérica al contar la frecuencia de cada palabra en el corpus, sin tomar en cuenta su importancia o contexto. Es útil para identificar los términos más frecuentes en un conjunto de datos, pero puede dar un peso excesivo a palabras muy comunes, lo que podría ocultar información relevante. En la *Figura 6* se observa que términos como *"error"*, *"client"*, *"not"* y *"server"* dominan visualmente y, por tanto, son las palabras más frecuentes en los logs. Estos resultados podrían indicar que los registros están dominados por errores relacionados con los servidores. Además, palabras como *"invalid"*, *"type"* y *"value"* apuntan a problemas frecuentes de validación y formatos de datos. Sin embargo, debido a que *Bag of Words* prioriza únicamente la frecuencia, algunos términos comunes como *"not"* y *"type"* pueden aparecer de forma prominente sin necesariamente aportar mucho valor contextual.

Figura 7. Nube de palabras con tf-idf

Fuente: Elaboración propia

TF-IDF (Term Frequency-Inverse Document Frequency) es una técnica que asigna pesos a las palabras en función de su frecuencia en un documento y su rareza en todo el corpus. A diferencia de *Bag of Words*, *TF-IDF* reduce la importancia de términos comunes y resalta palabras que son menos frecuentes, pero más informativas en el conjunto de documentos. En la *Figura 7*, se observan nuevamente términos clave como *"error"*, *"client"*, *"server"* y *"not"*, pero esta vez con un peso más equilibrado y menos predominancia de palabras extremadamente frecuentes. Palabras como *"forbidden"*, *"gateway"* y *"validation"* toman mayor relevancia, indicando presencia de errores específicos relacionados con accesos denegados y problemas de validación.

5.3. Ajuste de modelo no supervisado

La fase de modelamiento tiene como objetivo descubrir patrones subyacentes y relaciones significativas dentro de los logs de las APIs de TABi Connect, utilizando técnicas avanzadas de Machine Learning no supervisado. Dado el volumen y la complejidad de los datos generados por las APIs, se implementaron diversos enfoques y modelos para tratar tanto los datos estructurados como los no estructurados, como los mensajes en lenguaje natural presentes en los logs.

En primer lugar, se empleó el modelo LDA (Latent Dirichlet Allocation) para el modelado de tópicos, una técnica popular en el análisis de texto. Para optimizar este modelo, se configuraron varios parámetros clave, como el uso de CountVectorizer con filtros de frecuencia, donde los parámetros min_df = 0.01 y max_df = 0.95 permitieron trabajar con términos que aparecen en un

rango de frecuencia adecuado, excluyendo palabras demasiado raras o comunes. Además, se experimentó con diferentes combinaciones de n-gramas para capturar tanto patrones de palabras individuales como secuencias de palabras más complejas. También se ajustaron varios valores para los hiperparámetros alpha y beta, que controlan la distribución de los tópicos y las palabras dentro de esos tópicos, respectivamente. Para evaluar la efectividad de los modelos, se utilizaron métricas de evaluación como perplexity y coherence, que proporcionan una indicación de la calidad de los modelos generados, y visualización e interpretación humana. Para la visualización y mejor interpretación de los resultados, se utilizó la herramienta pyLDAvis, que permite explorar la distribución de los tópicos y su relación con las palabras clave.

Además del modelado de tópicos, se implementó un enfoque de clustering para identificar agrupaciones en los datos que podrían indicar patrones o anomalías recurrentes en los logs. Este proceso implicó trabajar con varias representaciones de los datos de texto. Primero, se dummizaron las variables categóricas clave, como type, api, status code y summary, con el fin de convertirlas en variables numéricas aptas para los algoritmos de clustering. Para la variable detail, que contiene mensajes en lenguaje natural, se probaron diferentes técnicas de vectorización de texto, como CountVectorizer, TF-IDF y Word2Vec. El CountVectorizer y el TF-IDF son enfoques tradicionales de representación de texto, mientras que Word2Vec ofrece una representación más avanzada que captura relaciones semánticas entre las palabras. Estos métodos permitieron transformar los datos textuales en vectores numéricos, facilitando su uso en los modelos de clustering. A su vez, se exploró una variante de cada uno de estos enfoques utilizando PCA (Análisis de Componentes Principales) para la reducción de dimensionalidad. El uso de PCA buscaba optimizar los algoritmos de clustering al reducir la complejidad de los datos sin perder la información relevante, mejorando así la interpretabilidad y el rendimiento de los modelos.

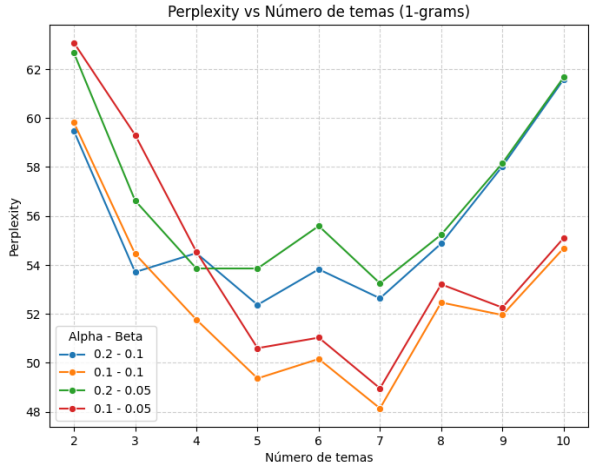
5.3.1. LDA

5.3.1.1. 1-grams

En este apartado, se implementa LDA (Latent Dirichlet Allocation) utilizando 1-grams para identificar temas latentes dentro del texto. Para optimizar el rendimiento del modelo, se evaluaron diferentes combinaciones de los hiperparámetros Alpha y Beta, y se compararon distintas cantidades de temas, midiendo el desempeño mediante la métrica de perplejidad. La *Figura* 8

muestra la evolución de la perplejidad en función del número de temas y revela la combinación de parámetros que ofrece el mejor balance entre ajuste del modelo y complejidad.

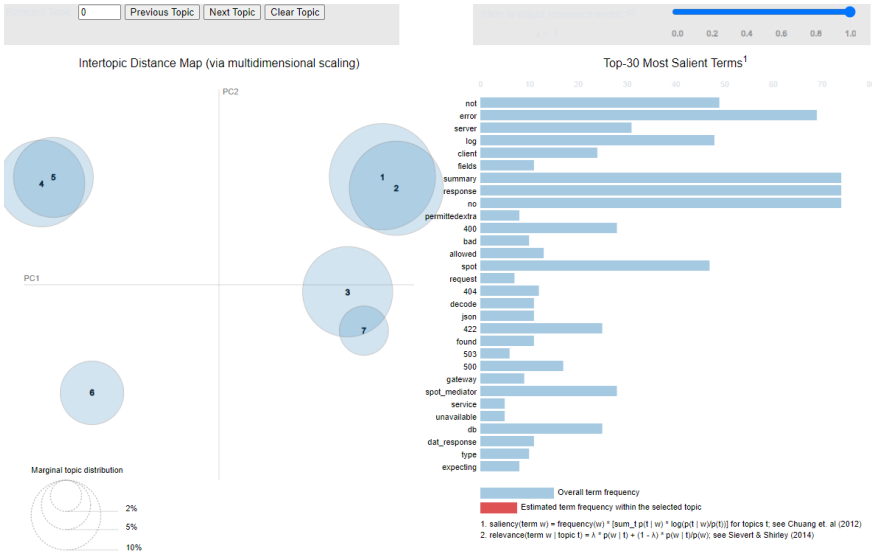
Figura 8. Validación cruzada - Perplexity



Fuente: Elaboración propia

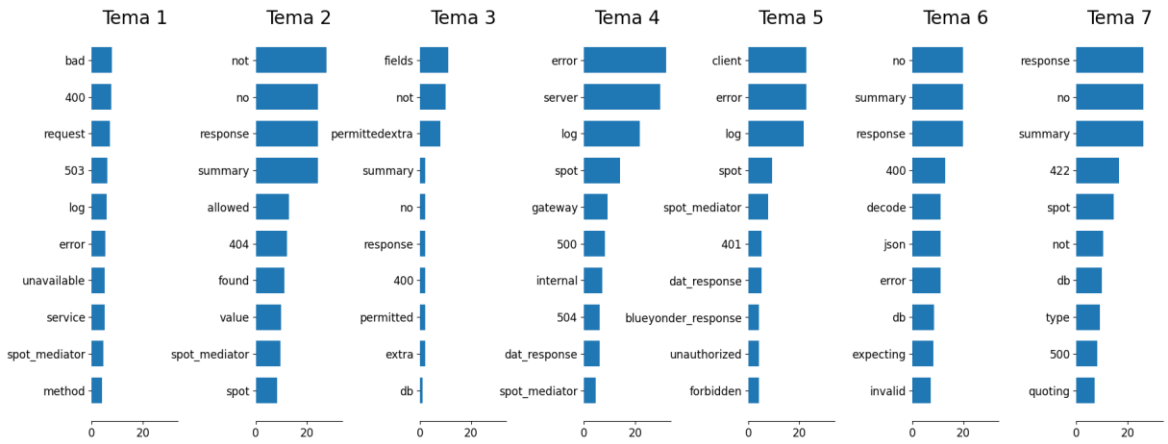
Como se evidencia en la *Figura 8*, la configuración de hiperparámetros Alpha = 0.1 y Beta = 0.1 logra el mejor rendimiento del modelo LDA, alcanzando la menor perplejidad con 5 a 7 temas. Esto indica que en este rango se obtiene un equilibrio óptimo entre ajuste y complejidad del modelo. A medida que el número de temas supera este punto, la perplejidad comienza a aumentar, sugiriendo que modelos con más de 7 temas no aportan mejoras significativas en la asignación de los temas. Los resultados de la mejor combinación de parámetros según la Perplexity se muestran en la *Figura 9*, donde se muestra la distribución y superposición de los temas; y en la *Figura 10*, que representa el top 10 de palabras más representativas de cada tema. Este modelo resulta en 7 temas que no están bien diferenciados, ya que se superponen en el espacio mostrado en la *Figura 9* y comparten varias palabras unos temas con otros como se evidencia en la *Figura 10*.

Figura 9. Distribución de temas LDA – Mejores parámetros según la Perplexity



Fuente: Elaboración propia

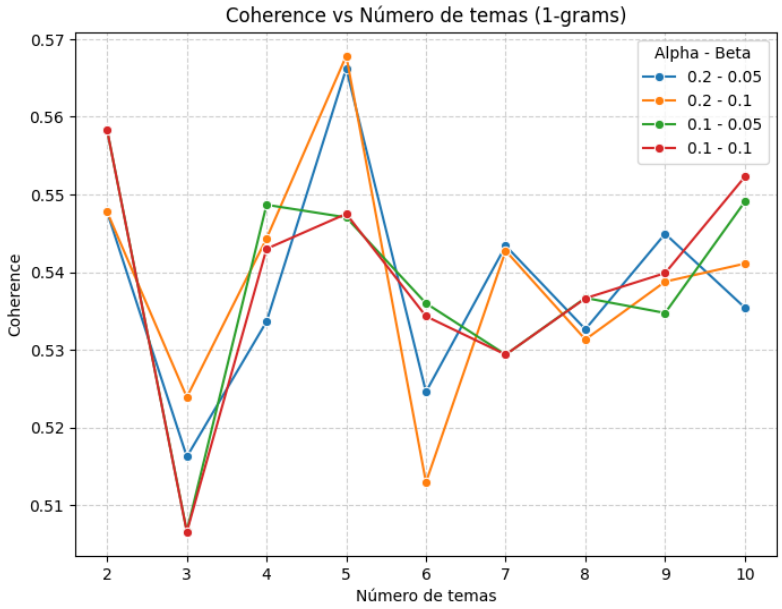
Figura 10. Top 10 palabras LDA - Mejores parámetros según la Perplexity



Fuente: Elaboración propia

Similar al punto anterior, se evaluaron diferentes combinaciones de los hiperparámetros Alpha y Beta, y se compararon distintas cantidades de temas, midiendo el desempeño mediante la métrica de coherencia. La *Figura 11* muestra la evolución de la coherencia en función del número de temas y permite saber la mejor combinación de parámetros respecto a esta métrica.

Figura 11. Validación cruzada - Coherence

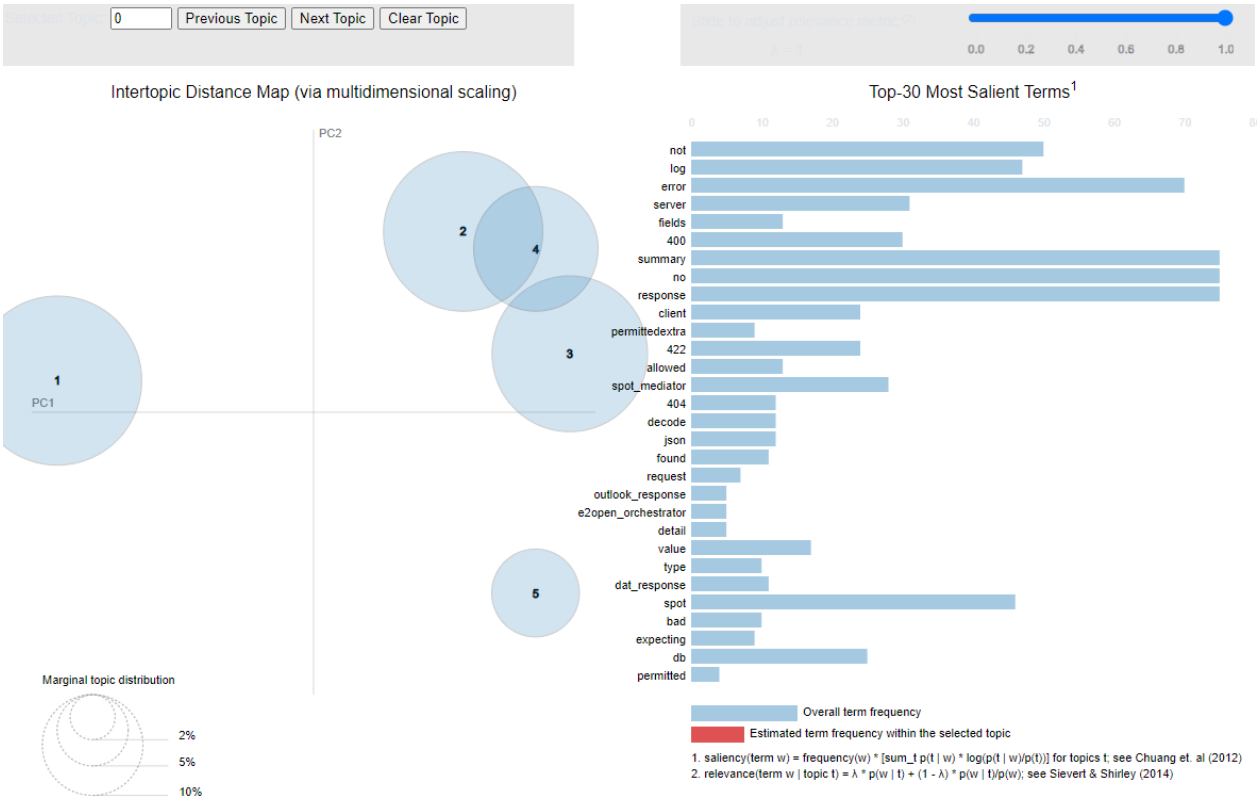


Fuente: Elaboración propia

Las combinaciones de hiperparámetros Alpha = 0.2 - Beta = 0.1 y Alpha = 0.2 - Beta = 0.05 presentan el mejor desempeño del modelo LDA, logrando la mayor coherencia cuando se definen 5 temas. Esto indica que, en este rango, se alcanza un balance adecuado entre la capacidad del modelo para ajustarse a los datos y su complejidad. Sin embargo, al incrementar el número de temas más allá de este punto, la coherencia comienza a disminuir, mostrando que más de 5 temas no proporciona mejoras significativas en la identificación y asignación de estos. Los resultados de la mejor combinación de parámetros según la Coherence se muestran en la *Figura 12*, donde se muestra la distribución y superposición de los temas; y en la *Figura 13*, que representa el top 10 de palabras más representativas de cada tema.

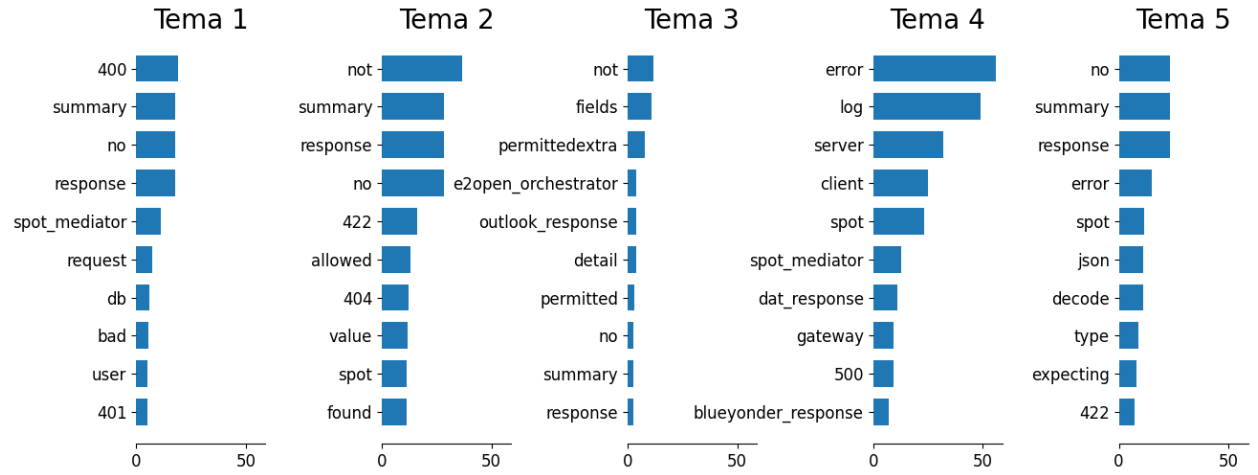
Similar a los resultados obtenidos usando la Perplexity como criterio, este modelo resulta en 5 temas que no están bien diferenciados, ya que se superponen o están muy cerca unos de otros en el espacio mostrado en la *Figura 12* y comparten varias palabras unos temas con otros como se evidencia en la *Figura 13*.

Figura 12. Distribución de temas LDA – Mejores parámetros según la Coherence



Fuente: Elaboración propia

Figura 13. Top 10 palabras LDA - Mejores parámetros según la Coherence



Fuente: Elaboración propia

5.3.1.2. N-grams

Debido a que los temas obtenidos en el apartado anterior utilizando 1-grams no presentaron interpretaciones claras o significativas, se decidió iterar sobre diferentes modelos LDA empleando diversas combinaciones de n-gramas. Para esta fase inicial, la evaluación de los modelos se llevó a cabo principalmente con la métrica de perplexity, ya que calcular la coherence no resulta adecuado cuando se incluyen n-gramas porque esta métrica está diseñada para palabras individuales y no para combinaciones de términos. En consecuencia, el proceso de revisión y evaluación de los temas fue en gran medida, manual, realizando un análisis detallado de las asignaciones hasta identificar una configuración que ofreciera una segmentación más coherente y significativa de los temas. Este enfoque permitió afinar gradualmente los resultados y obtener una mejor representación de las estructuras latentes en los datos.

En primer lugar, para cada rango de n-gramas, se hizo una validación cruzada para encontrar la mejor combinación (perplejidad más baja) de Alpha, Beta y número de temas, encontrando lo que se muestra en la *Tabla 3.* La idea fue utilizar estos resultados para iniciar la exploración; pero, como ya se mencionó, la selección del modelo con la mejor asignación de temas fue en mayor parte manualmente y basado en conocimiento del proceso.

Tabla 3. Validación cruzada para n-gramas - Perplexity

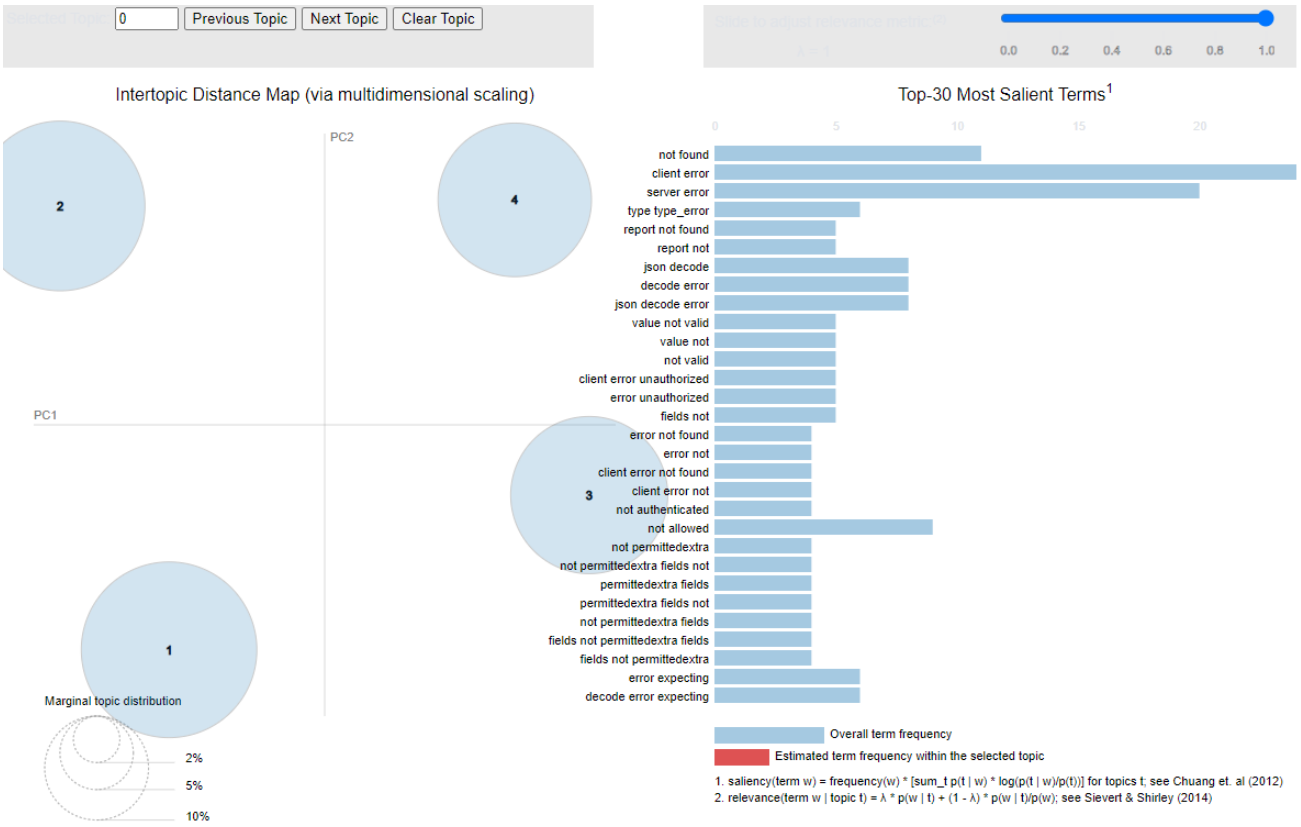
n_gram	alpha - beta	n_components	perplexity
(1, 1)	0,1 - 0,1	7	48,1430984
(1, 2)	0,1 - 0,1	9	105,828699
(1, 3)	0,1 - 0,05	10	158,702543
(1, 4)	0,1 - 0,05	10	209,590825
(2, 2)	0,1 - 0,1	7	85,545303
(2, 3)	0,1 - 0,1	8	143,015104
(2, 4)	0,1 - 0,05	9	179,699097
(3, 3)	0,1 - 0,1	6	115,227506
(3, 4)	0,1 - 0,05	10	151,790149
(4, 4)	0,1 - 0,05	10	87,4814043

Fuente: Elaboración propia

El modelo con los temas más interpretables fue un LDA con rango de n-gramas=(2,4), Alpha=0.1, Beta=0.01 y un total de 4 temas. La distribución de estos temas se puede visualizar en la *Figura 14* y el top 20 de palabras más representativas de cada tema se presenta en la *Figura 15*. Este modelo

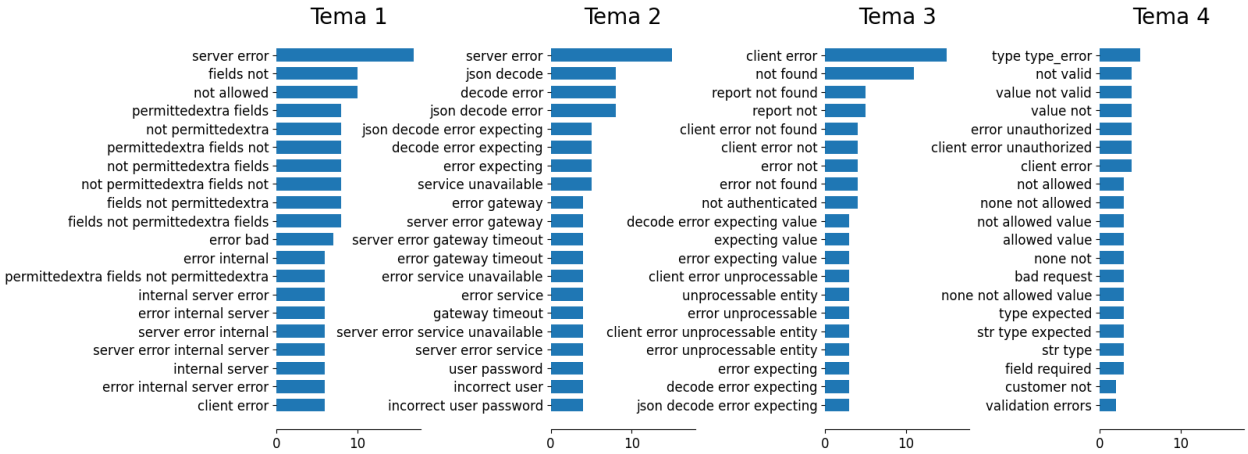
permite obtener 4 temas bastante bien diferenciados. Ya que, como se observa en la *Figura 14*, los temas están separados y no se superponen en el espacio; y, como lo muestra la *Figura 15*, los temas generados comparten muy pocos términos y combinación de términos.

Figura 14. Distribución de temas LDA – Mejor segmentación según evaluación manual



Fuente: Elaboración propia

Figura 15. Top 20 palabras LDA - Mejor segmentación según evaluación manual



Fuente: Elaboración propia

5.3.2. Clustering

En este apartado se ajustan modelos de clustering DBSCAN con las variables type, api, status code y summary dummizadas y la variable detail tokenizada con la técnica Bag of Words, TF-IDF y Word2Vec, escalada con Standard Scaler. En primer lugar, para cada técnica de tokenización, se reduce la dimensionalidad de los logs a 3 dimensiones aplicando PCA y se visualizan los puntos para determinar el tamaño de n-grams con el que se obtiene una distribución más adecuada y cómoda para algoritmos de clustering. Luego, se hace una validación cruzada para observar el rendimiento del modelo con diferentes combinaciones de los parámetros epsilon y min samples; y, de esta manera, seleccionar el modelo DBSCAN con la mejor agrupación según el Silhouette Score, Calinski harabasz score y conocimiento del proceso. Los valores iniciales de los hiperparámetros fueron seleccionados mediante técnicas basadas en la *distancia k-vecinos*. Posteriormente, se iteró de forma experimental para ajustar estos parámetros y refinar el modelo, garantizando que los clusters identificados fueran consistentes con el problema y la distribución de los datos.

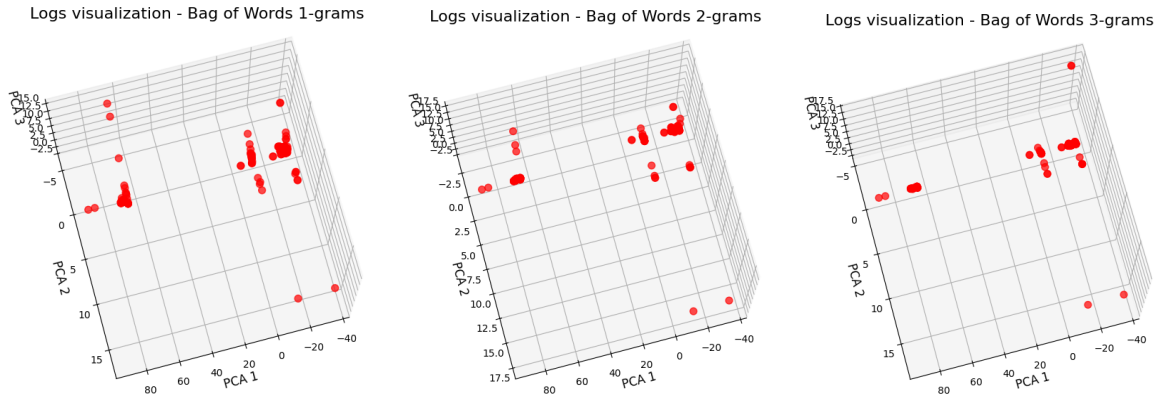
Se decidió utilizar DBSCAN en lugar de K-Means debido a que la distribución de los datos no cumple con los supuestos necesarios para el correcto desempeño de K-Means. En particular, los clusters presentes en los datos no tienen una forma esférica ni tamaños homogéneos, como lo requiere el algoritmo K-Means para minimizar la varianza interna de los grupos. Además, se observaron regiones de baja densidad y la presencia de valores atípicos que podrían distorsionar

los resultados de K-Means al influir en la posición de los centroides. Por el contrario, DBSCAN es más adecuado en este caso, ya que puede identificar clusters con formas arbitrarias basándose en la densidad local de los datos y excluir los puntos dispersos como ruido, lo que garantiza una segmentación más robusta y precisa.

5.3.2.1. DBSCAN

5.3.2.1.1. Bag of Words

Figura 16. Distribución de logs - Bag of Words



Fuente: Elaboración propia

Para los resultados obtenidos en la *Figura 16* se aplicó PCA para reducir la dimensión a 3 componentes, obteniendo como mínimo un 95% de la varianza explicada. Se eligió el modelo con 2-grams como la representación más adecuada debido a la distribución de los puntos mostrada en la imagen. A diferencia de los gráficos generados con 1-grams y 3-grams, la visualización de 2-grams evidencia una separación más clara y una estructura más compacta en los clusters potenciales.

Tabla 4. Validación cruzada - Bag of Words

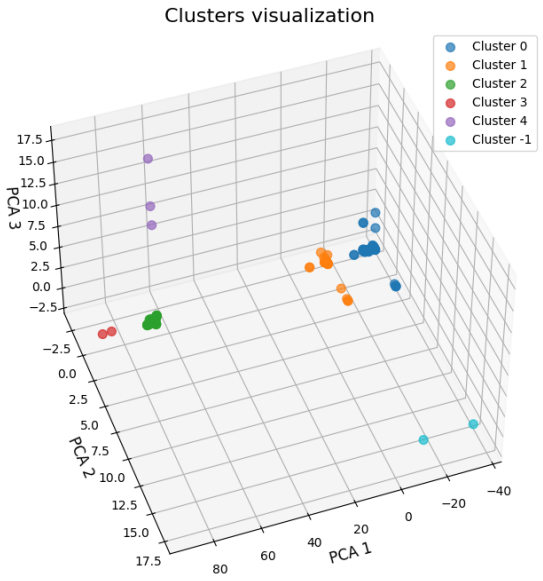
eps	min_samples	silhouette	calinski
5	2	0,907443	16347,29
6	2	0,907443	16347,29
7	2	0,900344	17974,78

7	3	0,900344	17974,78
7	5	0,900344	17974,78
7	10	0,900344	17974,78
5	3	0,8664	949,75
5	5	0,8664	949,75
5	10	0,8664	949,75
6	3	0,8664	949,75
6	5	0,8664	949,75

Fuente: Elaboración propia

Con base en los resultados observados en la *Tabla 4* obtenidos de la validación cruzada, se seleccionaron los valores de *eps = 7* y *min_samples = 2* como los mejores parámetros para el modelo DBSCAN. Esta decisión se fundamenta en que dichos valores ofrecen un equilibrio óptimo entre silhouette, Calinski-Harabasz y el ruido. Los resultados de los clusters se muestran en la *Figura 17*, donde se evidencia que los 5 clusters resultantes del modelo parecen estar bien definidos y que hay presencia de ruido en los logs.

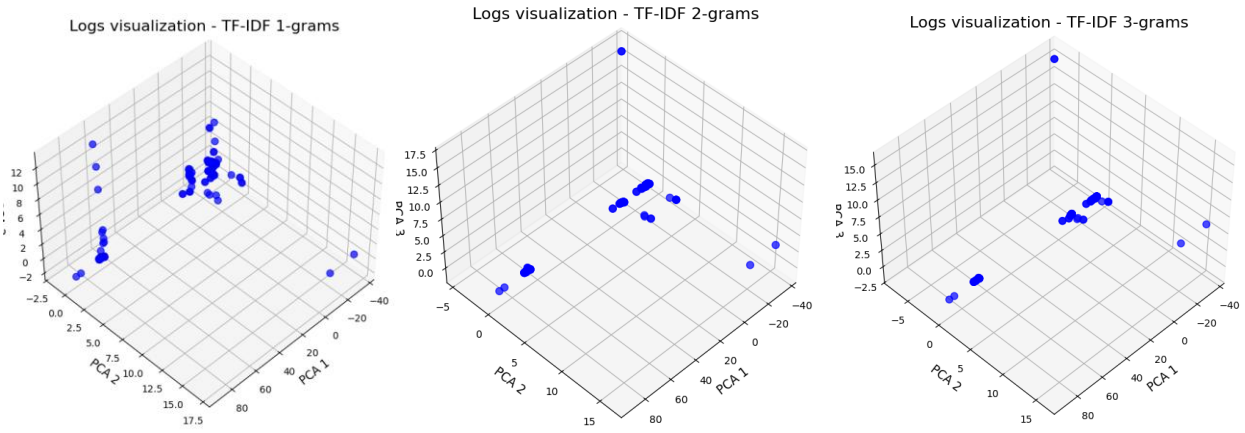
Figura 17. Resultado de clusters - Bag of Words



Fuente: Elaboración propia

5.3.2.1.2. TF-IDF

Figura 18. Distribución de logs - TF-IDF



Fuente: Elaboración propia

En la *Figura 18*, similar al punto anterior, se utilizó el método de PCA para reducir la dimensionalidad de los datos a 3 componentes principales para poder observar su distribución y asegurando que al menos el 95% de la varianza fuera preservada. Entre las representaciones analizadas, se seleccionó el modelo basado en trigramas (2-grams) como el más adecuado, dado que la distribución de los puntos en el gráfico muestra una mejor separación más evidente y una estructura más cohesiva entre los posibles clusters. En contraste, las representaciones obtenidas con unigramas (1-grams) y trigramas (3-grams) presentan distribuciones menos claras y con mayor dispersión, lo que dificulta la identificación de grupos bien definidos.

Tabla 5. Validación cruzada - TF-IDF

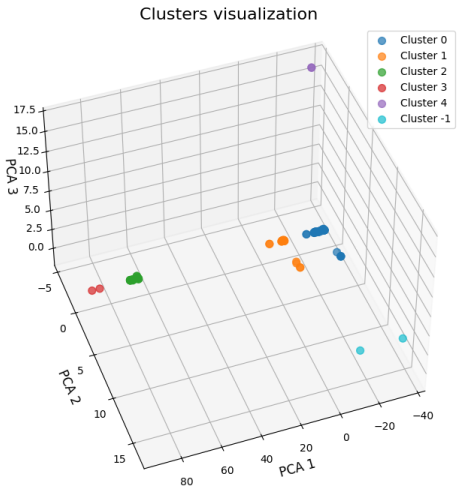
eps	min_samples	silhouette	calinski
5	2	0,907594	16343,99
6	2	0,907594	16343,99
7	3	0,900461	17965,75
7	5	0,900461	17965,75
8	3	0,900461	17965,75
8	5	0,900461	17965,75
9	3	0,900461	17965,75
9	5	0,900461	17965,75
7	2	0,900461	17965,75

8	2	0,900461	17965,75

Fuente: Elaboración propia

Teniendo en cuenta los resultados obtenidos en la *Tabla 5*, se determinaron los valores de eps = 7 y min_samples = 2 como los parámetros más adecuados para el modelo DBSCAN usando TF-IDF. Estos parámetros logran un equilibrio entre las métricas seleccionadas. Los resultados de los clusters se muestran en la *Figura 19*, donde se evidencia que los 5 clusters resultantes del modelo parecen estar definidos correctamente y que hay presencia de ruido en los logs, muy similar al modelo con Bag of Words.

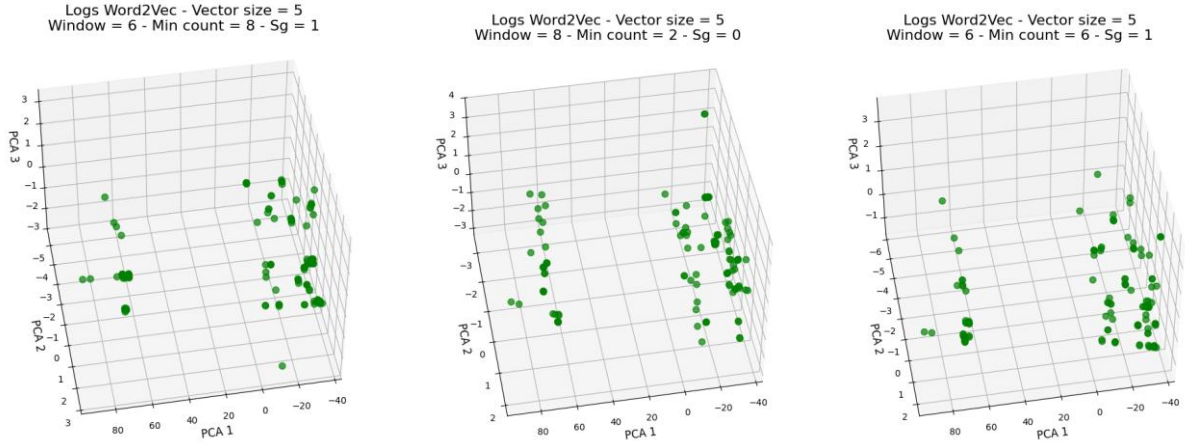
Figura 19. Resultado de clusters - TF-IDF



Fuente: Elaboración propia

5.3.2.1.3. Word2Vec

Figura 20. Distribución de logs - Word2Vec



Fuente: Elaboración propia

La distribución de la primera visualización de la *Figura 20* parece ser la más adecuada. Los puntos en este gráfico muestran una separación más evidente y una estructura más diferenciada en comparación con las otras distribuciones. Las visualizaciones del centro y la derecha presentan mayor solapamiento entre los puntos y una dispersión menos estructurada, lo que dificultaría la formación de clusters bien definidos.

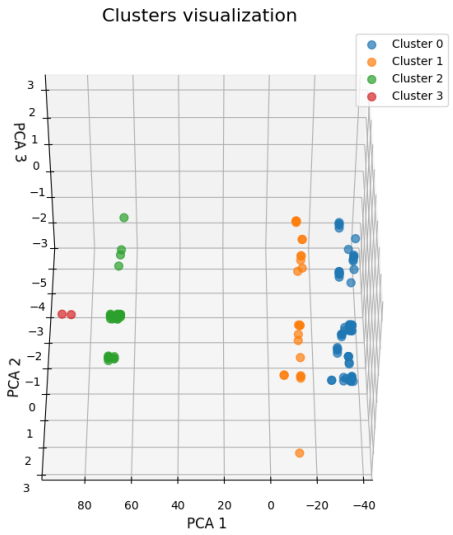
Tabla 6. Validación cruzada - Word2Vec

eps	min_samples	silhouette	calinski
5	2	0,907988	16448,16
6	2	0,907988	16448,16
7	2	0,90097	18062,70
7	3	0,90097	18062,70
7	5	0,90097	18062,70
8	2	0,90097	18062,70
8	3	0,90097	18062,70
8	5	0,90097	18062,70
9	2	0,90097	18062,70
9	3	0,90097	18062,70

Fuente: Elaboración propia

Basados en los resultados de la *Tabla 6*, se determinaron los valores de eps = 7 y min_samples = 2 como los parámetros más adecuados para el modelo DBSCAN. Esta elección logran un equilibrio entre las métricas de silhouette, Calinski-Harabasz y la cantidad de ruido. Los resultados de los clusters se muestran en la *Figura 21*, donde se evidencia que los 4 clusters resultantes del modelo parecen estar bien definidos y que hay no presencia de ruido en los logs. Visualmente, parece ser esta la mejor técnica para tokenizar los mensajes de los logs.

Figura 21. Resultado de clusters - Word2Vec



Fuente: Elaboración propia

5.4. Análisis y evaluación del modelo

Esta etapa permite determinar la efectividad y la fiabilidad de los algoritmos implementados para analizar los logs de las APIs de TABi Connect. En esta sección, se presentan los criterios y métricas utilizados para medir el rendimiento del modelado de tópicos y de clustering. A través de un análisis cuantitativo y cualitativo, se busca validar que las técnicas aplicadas logren identificar patrones que sean coherentes y útiles en el contexto del negocio.

Para la evaluación del modelo LDA (Latent Dirichlet Allocation), se emplearon las métricas de perplexity, coherence (en caso de no usar n-gramas) y conocimiento del negocio. La perplexity mide qué tan bien el modelo generaliza al predecir documentos no observados, mientras que la coherence evalúa la calidad semántica de los temas generados, asegurando su interpretabilidad.

Estas métricas son analizadas de forma complementaria, buscando un equilibrio entre rendimiento matemático y relevancia contextual.

En el caso de los modelos de clustering, específicamente DBSCAN, se utilizan métricas como el Silhouette Score, que evalúa la separación y cohesión de los clusters, y el Calinski-Harabasz Index, que mide la relación entre la dispersión interna y externa de los grupos.

5.4.1. DBSCAN

Tabla 7. Resultado métricas para Clustering

Técnica	eps	min_samples	silhouette	calinski	clusters
Bag of Words	7	2	0,900344	17974,7	5
TF-IDF	7	3	0,900461	17965,7	5
Word2Vec	7	2	0,90097	18062,7	4

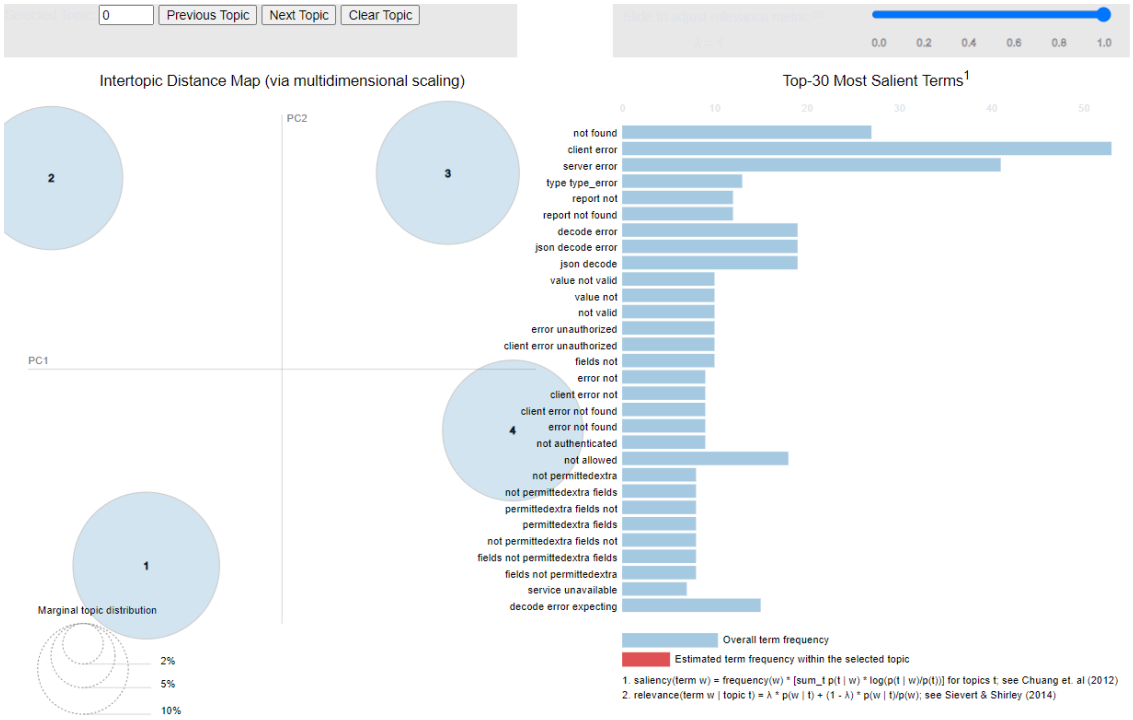
Fuente: Elaboración propia

De acuerdo con los resultados de la sección *Tabla 7*, los modelos de clustering implementados con DBSCAN muestran un buen rendimiento en términos de las métricas de validación. Los valores cercanos a 0.9 en el Silhouette Score indican que los clusters están bien definidos y separados. Los altos valores observados Calinski-Harabasz Index confirman que los clusters presentan una buena relación entre la cohesión interna y la separación externa.
Sin embargo, a pesar de su buen rendimiento técnico, el análisis de los clusters con conocimiento del negocio no muestra patrones coherentes ni interpretables.

5.4.2. LDA

El modelo con los temas más interpretables fue un LDA con rango de n-gramas=(2,4), Alpha=0.1, Beta=0.05 y un total de 4 temas. La perplexity con este modelo es de 176, que es más alta que en otras combinaciones de parámetros, pero para este modelo se le dio bastante importancia a la interpretabilidad de los temas usando conocimiento del proceso.

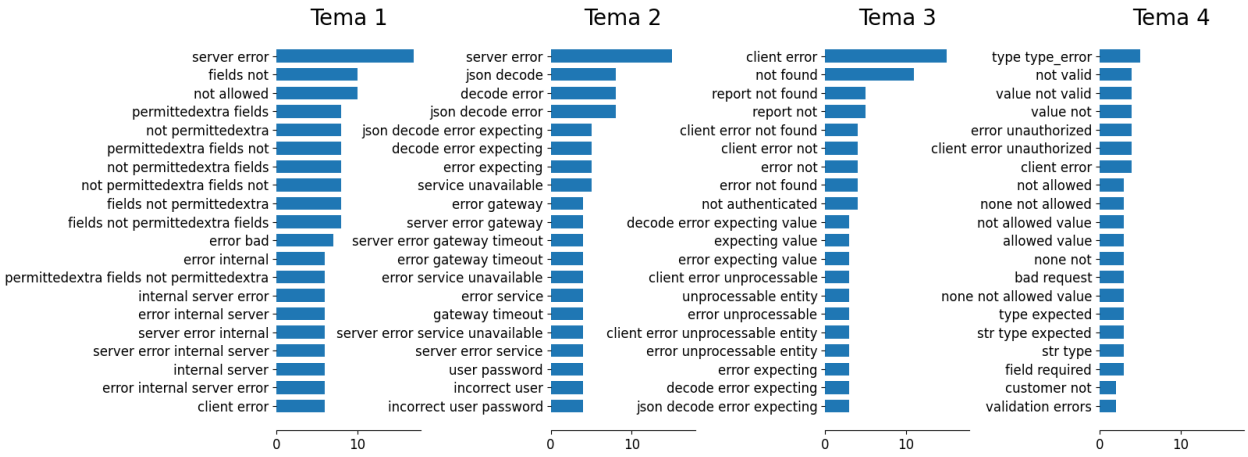
Figura 22. Resultados LDA - PyLDAvis



Fuente: Elaboración propia

Los temas mostrados en la *Figura 22* están razonablemente separados, lo que indica que el modelo ha capturado diferencias claras entre ellos. El tamaño de los círculos es similar, lo que indica que los 4 temas cubren proporciones similares del corpus.

Figura 23. Resultados LDA – Términos más relevantes por tema



Fuente: Elaboración propia

En la *Figura 23* se evidencian los 20 términos o combinación de términos más representativos de cada tema. El Tema 1 refleja errores y excepciones generales, como "server error" o "internal server error". Estos errores están relacionados con problemas de validación de datos, como los términos "fields not permitted" y "permittedextra fields"; este tema representa el resultado final del proceso, donde las solicitudes con datos incorrectos o fallas acumuladas desencadenan errores importantes. El Tema 2 agrupa errores internos durante el procesamiento de solicitudes, como fallos de decodificación ("json decode error") o servicios no disponibles ("service unavailable"); estos errores ocurren generalmente cuando las APIs externas no responden adecuadamente o envían datos inesperados. En el Tema 3, los errores se enfocan en problemas tanto del lado del cliente como del servidor, incluyendo "client error not found" y "error unprocessable entity"; este tema engloba errores HTTP 500 y 400, que se relacionan con interacciones problemáticas entre el cliente y la API. Por último, el Tema 4 identifica errores externos devueltos por las APIs, como "type type_error" y "value not valid"; estos errores reflejan problemas de validación en las solicitudes enviadas desde el sistema, indicando discrepancias en los formatos de datos o falta de alineación con los requisitos especificados por las APIs externas.

La relación entre los temas presenta un flujo progresivo y condicional de errores. El proceso se inicia en el Tema 4, donde el sistema envía solicitudes a APIs externas. Si estas solicitudes son válidas, el flujo continúa normalmente. Sin embargo, si hay problemas de validación o discrepancias en los datos enviados, se generan errores externos relacionados con datos inválidos o no permitidos, como "type type_error" o "value not valid". Si las solicitudes pasan exitosamente esta etapa y no se generan errores del Tema 4, el sistema procede al procesamiento interno. Aquí entra en juego el Tema 2, que aborda errores internos como "json decode error" o "service unavailable". Estos errores surgen cuando el sistema enfrenta dificultades para procesar datos recibidos de las APIs externas. Si no se producen errores en esta etapa, el flujo continúa sin interrupciones. Sin embargo, si ocurren fallos, estos errores internos se registran y afectan el desempeño del sistema. Finalmente, si el sistema logra superar las validaciones y el procesamiento interno sin errores de los Temas 4 y 2, el flujo continúa hacia la siguiente etapa sin problemas. Pero, si surgen fallos más críticos que no fueron manejados adecuadamente, se registran errores generales descritos en el Tema 1, como "server error" o "internal server error".

A partir de este análisis, se han identificado y clasificado los errores en cuatro categorías principales, cada una representando un conjunto específico de fallas dentro del sistema:

- Tema 1: Errores de Validación de Datos
- Tema 2: Excepciones Internas en el Procesamiento
- Tema 3: Errores de Servidor y Respuestas Críticas
- Tema 4: Errores en Solicitudes Externas

Tabla 8. Representación de temas

Tema	Frecuencia
1	31,41%
2	22,31%
3	26,45%
4	19,83%

Fuente: Elaboración propia

De acuerdo con la *Tabla 8*, los Errores de Validación de Datos (31,41%) y los Errores de Servidor y Respuestas Críticas (26,45%) son los más recurrentes, evidenciando la necesidad de fortalecer las validaciones iniciales y optimizar las interacciones entre cliente y servidor. Aunque los Errores en Solicitudes Externas tienen la menor incidencia (19,83%), su ubicación al inicio del flujo los convierte en un punto muy importante para prevenir problemas posteriores. Tener en cuenta estas prioridades permitirá reducir significativamente los errores en cascada.

La jerarquización de errores permite identificar problemas en las primeras etapas del flujo, facilitando su resolución antes de que escalen a fallos críticos. Por ejemplo, los errores relacionados con validaciones de datos en el Tema 4 pueden detectarse y corregirse mediante procesos automatizados antes de que las solicitudes lleguen a APIs externas. Esto reduce significativamente la cantidad de errores subsiguientes y asegura que las interacciones iniciales sean exitosas.

El Tema 2, que abarca errores internos relacionados con el procesamiento de datos, puede ser monitoreado con sistemas automatizados que identifiquen patrones como "json decode error" o "service unavailable". Estos mecanismos no solo alertan sobre problemas en tiempos menores, sino que también permite llevar a cabo acciones correctivas de manera más rápida, reduciendo la intervención manual. Al priorizar el manejo de excepciones en esta etapa, el modelo asegura una

mejor gestión de las solicitudes, evitando que estos errores impacten en las siguientes fases del proceso.

6. Conclusiones y recomendaciones

Se logró tener un entendimiento y comprensión del proceso de monitoreo de las APIs que permitió diseñar una solución basada en Machine Learning No Supervisado para optimizar dicho monitoreo. Este proceso de monitoreo es muy manual, por lo que requiere de estrategias que permitan optimizar los recursos que se destinan para esto.

A pesar de las limitaciones debido al gran volumen de información, fue posible diseñar e implementar un pipeline que extraiga los logs desde CloudWatch usando la librería Boto3 de Python y que los limpie y transforme automáticamente para poder usarlos en ajustes de modelos de Machine Learning.

El modelo Latent Dirichlet Allocation (LDA) demostró ser efectivo para generar temas bien separados e interpretables en los logs. A pesar de que la métrica de perplexity indicó un desempeño matemático menor en comparación con otros modelos ajustados, el análisis desde el conocimiento del proceso permitió encontrar patrones coherentes. Este modelo identifica y clasifica los errores en cuatro temas principales: errores de validación de datos, excepciones internas en el procesamiento, errores de servidor y respuestas críticas y errores en solicitudes externas. Los resultados que se obtienen con este modelo hacen posible que sea necesario menos recursos como personal y tiempo para el monitoreo de las APIs, ya que permite obtener una categorización y priorización rápida respecto al flujo de las solicitudes.

En cuanto al modelo DBSCAN, se obtuvo un buen desempeño en términos de métricas como Silhouette Score y Calinski-Harabasz Index, lo que indica una adecuada separación y cohesión de los clusters. Sin embargo, desde el punto de vista del negocio, algunos patrones identificados por este modelo no resultaron completamente coherentes, evidenciando la necesidad de ajustes adicionales para mejorar sus resultados con los problemas específicos del sistema.

La jerarquización de errores basada en los temas permitió identificar que los errores de validación de datos (31,41%) y los errores de servidor (26,45%) son los más frecuentes, subrayando la necesidad de optimizar las validaciones iniciales y las interacciones cliente-servidor para minimizar problemas en cascada. Por otro lado, aunque los errores en solicitudes externas tienen una incidencia menor (19,83%), su ubicación al inicio del flujo los convierte en un aspecto relevante para prevenir fallas posteriores. Estos temas representan los diferentes puntos críticos en

el flujo de funcionamiento del sistema, destacando áreas donde se pueden realizar mejoras significativas.

El análisis destacó que una correcta jerarquización de los errores permite priorizar su resolución, evitando que problemas iniciales escalen a fallos críticos. Por ejemplo, errores relacionados con validaciones de datos en las primeras etapas del flujo pueden ser detectados y corregidos mediante procesos automatizados antes de que afecten a las APIs externas, reduciendo significativamente la cantidad de errores subsiguientes. Asimismo, la capacidad de automatizar la identificación de excepciones internas, como fallos en la decodificación de datos o servicios no disponibles, representa una oportunidad para mejorar la gestión de las APIs y reducir la intervención manual.

Se recomienda diseñar estrategias más robustas para manejar el creciente volumen de información generado por los logs, asegurando la escalabilidad y eficiencia del sistema a medida que aumenta la complejidad operativa. Por otro lado, explorar otras técnicas de Machine Learning y Deep Learning para encontrar patrones más complejos que permitan optimizar aún más el monitoreo de las APIs a través del análisis de logs. Si bien los resultados obtenidos en este proyecto son positivos, se requiere de patrones menos generales para lograr que el modelo logre un proceso de monitoreo de las APIs más automático o que requiera mínima intervención humana.

Referencias

Amazon AWS. (s.f.). *What is Amazon CloudWatch?* Obtenido de Amazon AWS: https://docs.aws.amazon.com/es_es/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html

Bertero, C., Roy, M., Sauvanaud, C., & Tredan, G. (2017). Experience Report: Log Mining Using Natural Language Processing and Application to Anomaly Detection. *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, (págs. 351-360). Toulouse, France. doi:https://doi.org/10.1109/ISSRE.2017.43

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

Egger, R. (2022). Text Representations and Word Embeddings: Vectorizing Textual Data. doi:10.1007/978-3-030-88389-8_16

Google LLC. (s.f.). *¿Qué es la inteligencia artificial (IA)?* Obtenido de https://cloud.google.com: https://cloud.google.com/learn/what-is-artificial-intelligence

He, P., Zhu, J., He, S., Li, J., & Lyu, M. R. (2016). An Evaluation Study on Log Parsing and Its Use in Log Mining. *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, (págs. 654-661). Toulouse, France. doi:10.1109/DSN.2016.66

Holdsworth, J., & Scapicchio, M. (Junio de 2024). *What is deep learning?* Obtenido de IBM: https://www.ibm.com/topics/deep-learning

Just, J. (Enero de 2024). Natural language processing for innovation search – Reviewing an emerging non-human innovation intermediary. *Technovation, 129*. doi:https://doi.org/10.1016/j.technovation.2023.102883

Koçi, R., Franch, X., Jovanovic, P., & Abelló, A. (2020). A Data-Driven Approach to Measure the Usability of Web APIs. *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, (págs. 64-71). Portoroz, Slovenia. doi:https://doi.org/10.1109/SEAA51224.2020.00021

Massahiro Shimaoka, A., Cordeiro Ferreira, R., & Goldman, A. (2024). The evolution of CRISP-DM for Data Science: Methods, Processes and Frameworks. *SBC Reviews on Computer Science*, 28–43. doi:https://doi.org/10.5753/reviews.2024.3757

Peng, J., Wu, W., Lockhart, B., Bian, S., Yan, J. N., Xu, L., . . . Wang, J. (2021). DataPrep.EDA: Task-Centric Exploratory Data Analysis for Statistical Modeling in Python. *In Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21)*, (págs. 2271–2280). New York, NY, USA. doi:https://doi.org/10.1145/3448016.3457330

Pinto Gurdiel, L., Morales Mediano, J., & Cifuentes Quintero, J. A. (2021). A Comparison Study Between Coherence and Perplexity for Determining the Number of Topics in Practitioners Interviews Analysis. 225-234.

Ramachandran, S., Agrahari, R., Mudgal, P., Bhilwaria, H., Long, G., & Kumar, A. (2023). Automated Log Classification Using Deep Learning. *Procedia Computer Science, 218*, Pages 1722-1732. doi:https://doi.org/10.1016/j.procs.2023.01.150

Ranjan, M., Barot, K. V., Vaishnavi, R., Anujaa, P., Shilpi, S., & Arif, S. (2023). Python: Empowering Data Science Applications and Research. *Journal of Operating Systems Development & Trends*, 27-33. doi:10.37591/joosdt.v10i1.576

Recabal Ávila, J. A. (2024). *Aplicación de técnicas de Big Data para el procesamiento de datos de la operación del sector eléctrico, 2024 [Tesis de pregrado, Universidad de Chile]*. Repositorio institucional. Obtenido de https://repositorio.uchile.cl/handle/2250/199959

Riadi, I., Istiyanto, J. E., Ashari, A., & Subanar. (2013). Log Analysis Techniques using Clustering in Network Forensics. *(IJCSIS) International Journal of Computer Science and Information Security*. doi:doi.org/10.48550/arXiv.1307.0072

Sandhu, A., & Mohammed, S. (2022). Detecting Anomalies in Logs by Combining NLP features with Embedding or TF-IDF. doi:10.36227/techrxiv.19498769.v1

Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research. *SN COMPUT. SCI, 2*. doi:https://doi.org/10.1007/s42979-021-00592-x

Sievert, C., & Shirley, K. (2014). LDAvis: A method for visualizing and interpreting topics. *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces* (págs. 63–70). Baltimore, Maryland, USA: Association for Computational Linguistics (ACL). doi:10.3115/v1/W14-3110

Wahyuningsih, T., & Chen, S. C. (2024). Analyzing Sentiment Trends and Patterns in Bitcoin-Related Tweets Using TF-IDF Vectorization and K-Means Clustering. *Journal of Current Research in Blockchain*, 48-69. doi:10.47738/jcrb.v1i1.11

Wang, J., Zhao, C., He, S., Gu, Y., Alfarraj, O., & Abugabah, A. (2022). LogUAD: Log unsupervised anomaly detection based on word2Vec. *Computer Systems Science and Engineering*, 1207-1222. doi:10.32604/csse.2022.022365

Zhou, Z., Qin, J., Xiang, X., Tan, Y., Liu, Q., & Xiong, N. N. (2020). News text topic clustering optimized method based on TF-IDF algorithm on Spark. *Computers, Materials & Continua*, 217-231. doi:10.32604/cmc.2020.06431