Tecnológico de Monterrey

Propuesta de Proyecto Hey, HERO

Datathon 2024

José Enrique Figueroa Rivera

David Fernando Betancourth Castellanos

Paula María Sánchez Morales

Ricardo Ruiz Almanza

5 de mayo de 2024

Tec de Monterrey

Hey, banco

Resumen

A lo largo de este documento se abordará la iniciativa tomada a partir de la base de datos proporcionada por el banco. En este caso se tomó como punto de partida la base de datos. En la red social Twitter había comentarios al banco y la plataforma. Dentro de estos comentarios, se encontraban tanto opiniones de gratificación o positivismo hacia el banco, como también, opiniones de negación y neutrales acerca de algún producto o servicio. El primer paso fue dividir la cantidad de comentarios positivos y los negativos.

Una vez encontrado esto, se procedió con una lluvia de ideas para llegar a la iniciativa deseada. Tras discusión interna del equipo, se llegó a la conclusión de que la mejor iniciativa, tomando como base los datos proporcionados, sería la creación de una inteligencia artificial que realice respuestas de manera automática dentro de las redes sociales por medio de los servidores de HeyFPT.

Nuestros objetivos con esta propuesta son la creación de un sistema on atención personalizada hacia cada uno de los usuarios, tener una disponibilidad de respuesta 24/7, optimizar el tiempo de respuesta y mejorar la disponibilidad de educación financiera al consumidor.

Índice

Índice de Figuras

Introducción

Este documento examina una iniciativa estratégica desarrollada a partir del análisis de datos obtenidos de una base de datos proporcionada por hey banco. La investigación se centró en la evaluación de comentarios registrados en la red social Twitter dirigidos al banco y su plataforma, abordando una amplia gama de percepciones que van desde la satisfacción y el elogio hasta la crítica y la neutralidad respecto a los productos y servicios ofrecidos.

Para llevar a cabo este análisis, se implementaron técnicas de análisis de datos y se desarrolló código especializado para la extracción, procesamiento y visualización de la información contenida en los comentarios. Además, se realizaron gráficas y visualizaciones para examinar patrones y tendencias dentro de los datos recopilados, permitiendo una comprensión más profunda de la percepción del público hacia el banco.

El primer paso de este estudio fue clasificar los comentarios en positivos, negativos y neutrales, lo que proporcionó una seria de gráficas y documentación con datos que nos permitieron profundizar y hacer una propuesta sobre alguna problemática dentro del banco.

Tras encontrar la problemática, se realizó una conexión entre propuestas financieras con propuestas educativas y de valor para el cliente. Para finalmente realizar la propuesta a el banco.

Marco Teórico

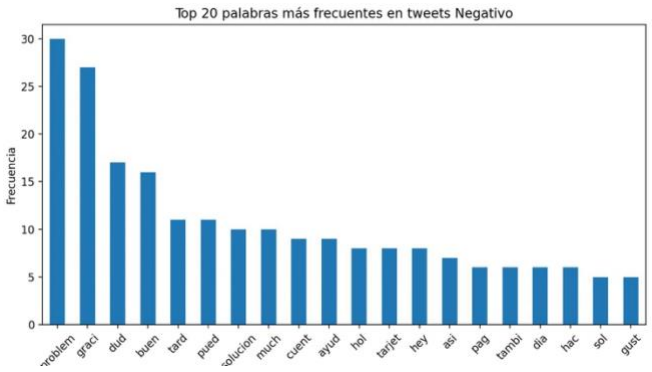
Como se mencionó anteriormente, se nos proporcionó una base de datos como punto de partida para realizar nuestra propuesta. Sin embargo, la base de datos por sí sola no es de mucha utilidad, por lo que debemos encontrar dentro de ella diversos factores que consideremos importantes. Primero que nada, observamos que la base de datos se dividía en tres columnas principales, siendo estas: Fecha, hora y comentario. Decidimos enfocarnos en maximizar el uso de los comentarios para poder entender mejor como es la reacción del público basándonos en estos.

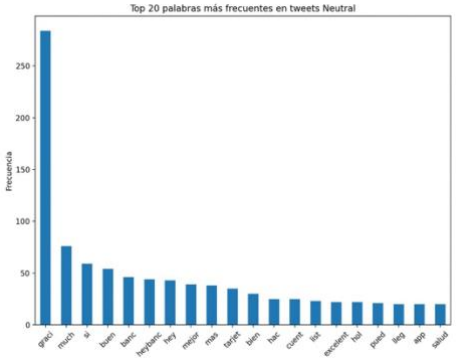
Para esto, fue necesario utilizar un código en Python para filtrar los comentarios que hayan tenido una connotación positiva, negativa o en su caso, neutral. Primero que nada, se realizó un código como el que se muestra en la Figura 1. que realiza un preprocesamiento de datos que realiza una tokenizacion de todas las palabras dentro de los comentarios. Esto nos es útil debido a que un texto como el que se proporciono requiere de librerías distintas para poder inducirlo directamente en el código final

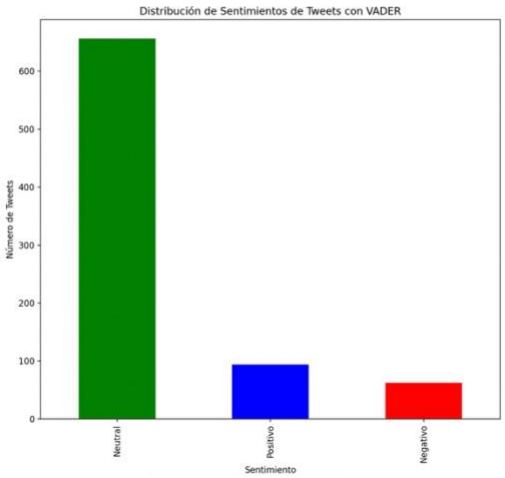
Por consiguiente, se inserta el documento ya tokenizado dentro de un segundo código, visto en la Figura 2, que elimina todos aquellos caracteres especiales que son irrelevantes para nuestro análisis, como lo son los acentos, así como la transformación de todas aquellas palabras que se encuentran en mayúsculas a minúsculas, como también la eliminación de stopwords. Para finalizar con una simplificación del texto por miedo de stemming, para terminar con un texto completamente tokenizado y limpio para inducir en el siguiente código.

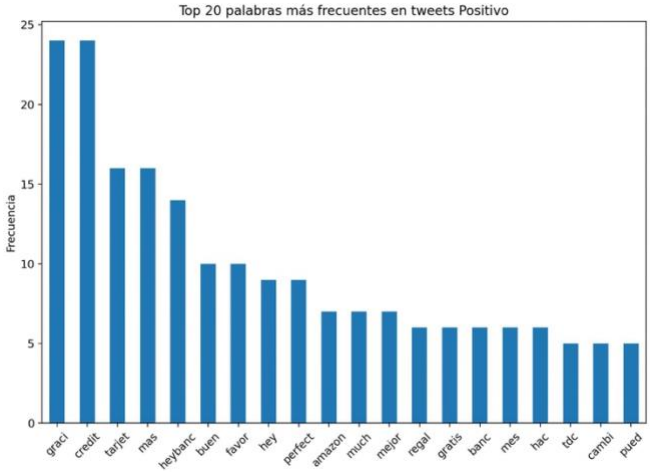
El siguiente código, visto en Figura 3, abarca la seccionización de la nueva base de datos dentro de tres principales secciones: positivo, negativo y neutral. En este caso se utiliza la herramienta de VADER (Valance Aware Dictionary and sEntiment Reasoner). Esta herramienta se encarga de identificar el sentimiento con el cual se está expresando el comentario y lo separa dentro de las tres secciones mencionadas anteriormente dependiendo del caso en el que se encuentre.

Una vez dividido el código en las tres secciones principales, el siguiente paso es identificar las palabras más comunes que se utilizan respectivamente. Por lo que se realiza una visualización de todas estas palabras y se realizan gráficas y recuentos sobre las mismas. Por lo que terminamos con gráficas, recuento del total de palabras comúnmente repetidas en la base de datos y todo esto seccionado dependiendo de si el comentario es positivo, negativo o neutral.



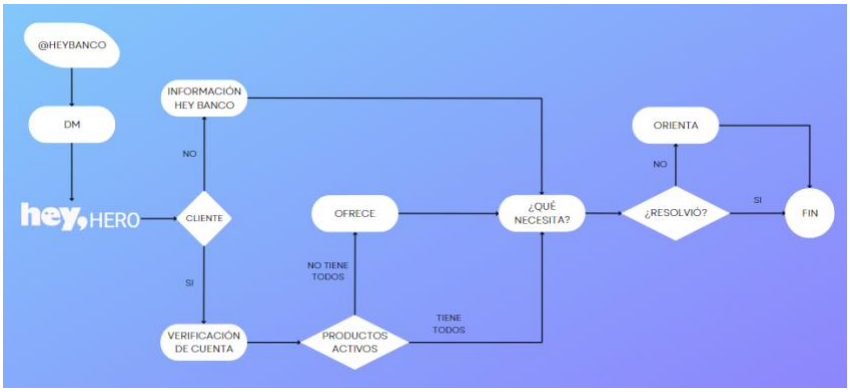






Una vez obtenido esto, precederemos con la lluvia de ideas para la realización de nuestra propuesta hacia el banco. En esta etapa, se realizó una discusión interna sobre las posibilidades de propuesta que podíamos tener. Para después asistir a el taller impartido e iluminarnos para poder pensar en nuestra propuesta final. En la cual decidimos realizar una metodología de respuesta automatizada a base de inteligencia artificial, tomando como punto de partida que la base de datos proporcionada es una extracción de la red social Twitter. Lo decidimos llamar HERO.

Esto funcionará de la siguiente manera:



El proyecto HERO by Hey (Hey Educa, Hey Resuelve, Hey Orienta) se concibe como una innovadora iniciativa para potenciar la asistencia y el servicio al cliente a través de las redes

sociales, con un enfoque especialmente dirigido hacia jóvenes adultos, comprendidos entre los 18 y 35 años de edad. La premisa central de este proyecto radica en proporcionar apoyo y asesoría a esta demografía mediante plataformas de redes sociales, donde suelen interactuar de manera habitual.

Este proyecto se basa en la implementación de una inteligencia artificial mejorada, una evolución del AI utilizado por Hey Banco conocido como HeyFPT, con el objetivo de dotarla de una mayor capacidad de respuesta y personalización. La automatización de respuestas hacia los clientes permitirá ofrecer una atención más rápida y efectiva, ya sea brindando una respuesta personalizada o ayudando a resolver problemas de manera ágil.

Cuando un usuario emita un comentario negativo hacia Hey Banco en las redes sociales, se activará una respuesta automática que invitará al usuario a enviar un mensaje directo (DM). A partir de ese momento, se desencadenará una conversación que buscará comprender y abordar las necesidades del cliente de manera individualizada.

Para los clientes de Hey Banco, la conversación comenzará indagando sobre su estado como cliente y, en caso afirmativo, solicitando su número de cuenta para poder atender su consulta específica. En caso de que el usuario no sea cliente, se procederá a preguntar directamente cuál es su problema para poder brindar la asistencia adecuada.

Además, se integrará una propuesta complementaria llamada HeyFuturo, que tiene como objetivo fomentar la cultura de la inversión y destacar la importancia de establecer una base para el retiro de cada individuo. Este proyecto busca crear un fondo de inversión a largo plazo, denominado Hey Futuro, que utilizará el cashback generado mensualmente para ser invertido, brindando así una oportunidad de crecimiento financiero a los usuarios.

Teniendo como propuestas complementarias la creación de una inteligencia artificial propia para tener un mayor control sobre la actualización y continua modernización de los servidores. Asimismo, crear una campaña informativa que abarque el cómo se puede utilizar de forma correcta la inteligencia y como se puede aprender acerca de finanzas al utilizar este chat de las redes sociales. Para finalizar con la propuesta de invertir en incrementar el personal para un avance y mejora continúa acelerada.

Código

```python
import pandas as pd
from preprocess import preprocess_text
from analyze_sentiment import get_vader_sentiment
from visual_data import plot_sentiment_counts, basic_extract_keywords, plot_words_sentiment

# Rutas de archivos
file_path = 'DB/heyDB.csv'
output_file_path = 'DB/processed_data.csv'

def main():
    """
    Función principal que procesa un archivo CSV de tweets, realiza análisis de sentimiento y genera visualizaciones.
    """
    # Cargar el archivo CSV en un DataFrame
    df = pd.read_csv(file_path)

    # Aplicar preprocesamiento de texto a cada tweet en una nueva columna
    df['processed_tweet'] = df['tweet'].apply(preprocess_text)

    # Aplicar análisis de sentimiento utilizando VADER a cada tweet procesado
    df['sentiment'] = df['processed_tweet'].apply(get_vader_sentiment)

    # Extraer palabras clave básicas de cada tweet procesado
    df['basic_keywords'] = df['processed_tweet'].apply(basic_extract_keywords)

    # Guardar el DataFrame modificado en un nuevo archivo CSV
    df.to_csv(output_file_path, index=True)

    # Contar la cantidad de cada tipo de sentimiento
    sentiment_counts = df['sentiment'].value_counts()

    # Imprimir las cuentas de cada tipo de sentimiento
    print(sentiment_counts)

    # Crear y mostrar gráfico de conteo de sentimientos
    plot_sentiment_counts(sentiment_counts)

    # Crear y mostrar visualización de palabras y su sentimiento
    plot_words_sentiment(df)

    # Imprimir comentarios con sentimiento negativo
#    negative_comments = df[df['sentiment'] == 'Negativo']['tweet'].tolist()
#    print("Comentarios con Sentimiento Negativo:")
#    for comment in negative_comments:
#        print("- ", comment)

if __name__ == "__main__":
    main()
```

Figura 1 Código main

```python
import re
import unicodedata
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer

import nltk
nltk.download('stopwords')
nltk.download('punkt')

def remove_accents(text):
    """
    Elimina los acentos y caracteres diacríticos de un texto utilizando Unicode normalization.

    Parameters:
    text (str): Texto del cual se eliminarán los acentos.

    Returns:
    str: Texto sin acentos ni caracteres diacríticos.
    """
    normalized_text = unicodedata.normalize('NFKD', text)  # Normalizar el texto con Unicode
    ascii_text = normalized_text.encode('ascii', 'ignore').decode('utf-8', 'ignore')  # Convertir caracteres a ASCII eliminando acentos
    return ascii_text


def preprocess_text(text):
    """
    Realiza el preprocesamiento básico de un texto para su análisis de sentimientos.

    El preprocesamiento incluye:
    - Convertir el texto a minúsculas.
    - Eliminar acentos y caracteres especiales.
    - Tokenizar el texto en palabras.
    - Eliminar palabras de parada (stopwords) del idioma español.
    - Aplicar stemming (reducción a la forma base) a las palabras.

    Parameters:
    text (str): Texto que se va a preprocesar.

    Returns:
    str: Texto preprocesado y listo para análisis de sentimientos.
    """
    # Convertir texto a minúsculas y eliminar acentos
    text = remove_accents(text.lower())

    # Eliminar caracteres especiales y números usando expresiones regulares
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenización
    tokens = word_tokenize(text)

    # Eliminar stopwords del idioma español
    stop_words = set(stopwords.words('spanish'))
    filtered_tokens = [word for word in tokens if word not in stop_words]

    # Stemming (reducción a la forma base de las palabras) usando SnowballStemmer
    stemmer = SnowballStemmer('spanish')
    stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]

    # Unir tokens procesados en una cadena de texto
    processed_text = ' '.join(stemmed_tokens)
    return processed_text
```

Figura 2 Código preprocess_data

```python
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk

nltk.download('vader_lexicon')

def get_vader_sentiment(text):
    """
    Clasifica un texto en Positivo, Neutral o Negativo utilizando el analizador de sentimientos VADER.

    VADER (Valence Aware Dictionary and sEntiment Reasoner) es una herramienta de análisis de sentimientos
    incluida en NLTK que asigna puntuaciones de polaridad (positivo, neutral o negativo) a un texto.

    Parameters:
    text (str): Texto que se va a analizar para determinar su sentimiento.

    Returns:
    str: Categoría de sentimiento ('Positivo', 'Neutral' o 'Negativo').
    """
    # Inicializar el analizador de sentimientos VADER
    sid = SentimentIntensityAnalyzer()

    # Obtener los puntajes de polaridad del texto
    score = sid.polarity_scores(text)

    # Determinar la categoría de sentimiento en base al puntaje compuesto
    if score['compound'] >= 0.05:
        return 'Positivo'
    elif score['compound'] > -0.05 and score['compound'] < 0.05:
        return 'Neutral'
    else:
        return 'Negativo'


```

Figura 3 Código analyze_sentiment

```python
import matplotlib.pyplot as plt
import pandas as pd



# Graficar el conteo de sentimientos
def plot_sentiment_counts(sentiment_counts):
    """
    Grafica el conteo de cada tipo de sentimiento.

    Parameters:
    sentiment_counts (pd.Series): Una serie pandas que contiene el conteo de cada tipo de sentimiento.

    Returns:
    None (Muestra la gráfica directamente).
    """
    plt.figure(figsize=(8, 5))
    sentiment_counts.plot(kind='bar', color=['green', 'blue', 'red'])
    plt.title('Distribución de Sentimientos de Tweets con VADER')
    plt.xlabel('Sentimiento')
    plt.ylabel('Número de Tweets')
    plt.show()

def basic_extract_keywords(text):
    """
    Extrae palabras clave de un texto eliminando puntuación y seleccionando solo palabras alfabéticas.

    Parameters:
    text (str): Texto del cual se extraerán las palabras clave.

    Returns:
    list: Lista de palabras clave extraídas del texto.
    """
    words = text.lower().split()  # Dividir el texto en palabras
    words_filtered = [word for word in words if word.isalpha()]  # Mantener solo palabras alfabéticas
    return words_filtered


def plot_words_sentiment(df):
    """
    Visualiza las palabras más frecuentes para cada sentimiento en tweets.

    Parameters:
    df (pd.DataFrame): DataFrame que contiene columnas 'sentiment' y 'basic_keywords'.

    Returns:
    None (Muestra las gráficas directamente).
    """
    basic_keywords_by_sentiment = df.groupby('sentiment')['basic_keywords'].sum()

    # Contar frecuencias de palabras clave para cada sentimiento
    basic_keyword_counts = {sentiment: pd.Series(keywords).value_counts().head(20) for sentiment, keywords in basic_keywords_by_sentiment.items()}

    # Visualizar las palabras más frecuentes para cada sentimiento
    for sentiment, counts in basic_keyword_counts.items():
        plt.figure(figsize=(10, 5))
        counts.plot(kind='bar', title=f'Top 20 palabras más frecuentes en tweets {sentiment}')
        plt.ylabel('Frecuencia')
        plt.xlabel('Palabras')
        plt.xticks(rotation=45)
        plt.show()
```

Figura 4 Código visual_data

```python
import pandas as pd
import regex as re
from collections import Counter

def count_emojis(text):
    # Use regex to find all emoji characters based on Unicode properties
    emoji_pattern = re.compile(r"\p{Emoji}")
    emojis = emoji_pattern.findall(text)

    # Filter out any characters that are digits ('0'-'9')
    emojis = [emoji for emoji in emojis if not emoji.isdigit()]

    emoji_counter = Counter(emojis)
    return emoji_counter

# Ruta al archivo Excel
file_path = "DB/heyDB.csv"

# Leer el archivo Excel en un dataframe de pandas
df = pd.read_csv(file_path)

# Nombre de la columna donde deseas contar emojis (por ejemplo, "tweet")
columna_deseada = "tweet"

# Contador total de emojis encontrados en la columna deseada
emojis_contados_total = Counter()

# Iterar sobre cada fila de la columna deseada y contar emojis
for fila in df.index:
    texto_celda = str(df.loc[fila, columna_deseada])  # Convertir a string para asegurar que se pueda buscar emojis
    emojis_contados_celda = count_emojis(texto_celda)
    emojis_contados_total += emojis_contados_celda

# Ordenar los emojis de mayor a menor según su conteo
emojis_ordenados = emojis_contados_total.most_common()

# Imprimir el recuento total de emojis encontrados en la columna deseada (ordenados de mayor a menor)
for emoji, count in emojis_ordenados:
    print(f"{emoji}: {count}")

```

Figura 5 Código emoji_checker

Conclusión

En conclusión, el análisis detallado de la base de datos proporcionada fue fundamental para desarrollar nuestra propuesta. A través de la aplicación de técnicas de procesamiento de lenguaje natural y herramientas como VADER, logramos segmentar los comentarios en positivos, negativos y neutrales, lo que nos permitió identificar tendencias y patrones en la percepción del público hacia Hey Banco en las redes sociales.

La visualización de datos y el recuento de palabras comunes en cada segmento nos brindaron una comprensión más profunda de las opiniones expresadas. Esta información fue crucial para generar ideas y llegar a la conclusión de que la implementación de una inteligencia artificial mejorada, denominada HERO by Hey, sería la solución más eficaz para mejorar la asistencia y el servicio al cliente en las redes sociales.

El proyecto HERO by Hey se enfoca en proporcionar una atención personalizada y eficiente a través de plataformas de redes sociales, utilizando la tecnología AI para responder automáticamente a los comentarios y mensajes de los usuarios. Además, se integra una iniciativa complementaria, HeyFuturo, que busca promover la cultura de la inversión y brindar oportunidades de crecimiento financiero a los clientes.