



**Memoria de Capstone Project**

# **NLP y Sentiment Analysis en Breaking Bad**

**EL DESCENSO A LA MALDAD DE WALTER WHITE**

/ BOOTCAMP DATA ANALYTICS

/ CONVOCATORIA DE MARZO 2024

Maite Lizarraga Téllez

**IMMUNE Technology Institute**

Paseo de la Castellana 89,  
28046 Madrid



# OUTLINE

## **1. INTRODUCCIÓN**

Motivación, objetivos, metodología, organización de la memoria, etc.

## **2. ESTADO DEL ARTE**

Trabajos previos

## **3. DESARROLLO**

Descripción, herramientas, tecnologías, etc

## **4. PRUEBAS Y RESULTADOS**

Tests realizados, resultados, etc.

## **5. CONCLUSIONES Y FUTURO TRABAJO**

Conclusiones, resumen y futuras mejoras.

# 1. INTRODUCCIÓN

White, W. (2012). *No. You're not done. You're not done until I say you're done* [Quote]. En *Breaking Bad* (temporada 5, episodio 7). Sony Pictures Television.

## <MOTIVACIÓN>

En la icónica serie *Breaking Bad*, Walter White, un simple profesor de química que se transforma en un despiadado fabricante de metanfetaminas, se convierte en un símbolo complejo de ambición desmedida, moralidad quebrada y poder descontrolado.

A lo largo de la serie, su evolución de hombre común a criminal notorio ofrece una fascinante exploración de la naturaleza del mal. En su ascenso y caída, Walter muestra su progresiva desconexión de los valores que antes definían su identidad.

En su célebre frase "No. You're not done. You're not done until I say you're done" (White, 2012), extraída de la quinta temporada, se revela la culminación de su transformación: un hombre que ha abrazado por completo el poder absoluto, dispuesto a sacrificar todo a su alrededor en nombre de su propio legado.

Este proyecto examina el descenso de Walter White hacia la maldad, explorando tanto sus momentos positivos como negativos de su historia para dibujar la imagen de un antihéroe polidrico y tan complejo como la naturaleza humana.

## <OBJETIVOS>

### > OBJETIVO 1:

Modelizar el descenso a la maldad de Walter White, protagonista de la aclamada serie *Breaking Bad* mediante el análisis de sentimientos y Procesamiento de Lenguaje Natural (NLP).

### > OBJETIVO 2:

Aprender a extraer, organizar, limpiar, transformar, cargar y analizar grandes cantidades de datos no estructurados.

### > OBJETIVO 3:

Aprender sobre Procesamiento de Lenguaje Natural y Análisis de Sentimientos. Saber preprocesar la información y utilizar Lexicones.

### > OBJETIVO 4:

Aprender sobre técnicas de Clustering.

### > OBJETIVO 5:

Aprender a trabajar en equipo, establecer líneas de trabajo y completar cada parte en tiempo y forma.

## <METODOLOGÍA>

El Natural Language Processing (NLP) es un campo de la inteligencia artificial que permite a las máquinas entender, interpretar y generar lenguaje humano. Una de sus aplicaciones más populares es el análisis de sentimiento, que busca identificar las emociones o actitudes expresadas en un texto, como si es positivo, negativo o neutral.

Existen varias aproximaciones para realizar el análisis de sentimiento:

- **Enfoque basado en reglas y lexicones:** Utiliza reglas predefinidas y diccionarios de palabras con connotaciones positivas, negativas o neutras para clasificar el texto.
- **Machine Learning:** Aquí se entrenan modelos con datos etiquetados, aprendiendo patrones en el lenguaje para predecir el sentimiento de nuevos textos.
- **Deep Learning y redes neuronales:** Si hay tiempo, se puede explorar el uso de redes neuronales, que permiten un análisis más sofisticado al captar matices más complejos en los textos.

## <ORGANIZACIÓN>

La memoria se divide en varias partes: una introducción, un repaso por la historia del NLP desde sus inicios hasta la actualidad, una explicación detallada del trabajo realizado para este capstone, las pruebas realizadas y los resultados obtenidos en dichas pruebas y las conclusiones a las que hemos llegado, así como los futuros trabajos que podrían prolongar o completar el trabajo actual.

## <ANTES DE COMENZAR>

Antes de comenzar, nos gustaría introducir una consideración importante: los sentimientos de una persona no son equiparables a sus acciones, aunque ambas estén relacionadas. Es fundamental distinguir entre lo que una persona siente y lo que hace para no caer en explicaciones simplistas de fenómenos muy complejos.

Los sentimientos son estados internos, emocionales, que surgen automáticamente en respuesta a experiencias, pensamientos o situaciones. No siempre se tiene un control consciente sobre ellos.

A diferencia de los sentimientos, las acciones son algo sobre lo que generalmente tenemos más control, ya que son los comportamientos que elegimos llevar a cabo como respuesta a nuestros pensamientos, emociones o situaciones externas.

*“Sin embargo, la cuestión parece claramente más complicada, puesto que el miedo involucra también de algún modo la creencia en la existencia de un peligro, y la ira la creencia de que alguien nos ha causado un daño o ha intentado hacerlo. «Peligro» y «daño» son términos evaluativos. [...] Así, por ejemplo, la creencia de que algo es peligroso haría razonable y daría lugar al deseo de actuar con vistas a librarnos de ese algo, y este deseo, junto con la creencia de que (digamos) huir es el mejor modo de lograr ese objetivo, haría a su vez razonable la huida y daría lugar a ella” (López, 2021).*

Esta disonancia entre lo que un personaje siente y lo que hace puede ofrecer una perspectiva más profunda sobre su psicología y motivaciones.

## 2. ESTADO DEL ARTE

### <NATURAL LANGUAGE PROCESSING>

El procesamiento de lenguaje natural (NLP) es una de las áreas más dinámicas y populares dentro de la inteligencia artificial, con aplicaciones en diversos dominios tanto relacionados como distantes. Debido al constante incremento de investigaciones publicadas, resulta difícil capturar el estado actual del campo y facilitar la incorporación de nuevos investigadores.

En la literatura actual, la mayoría de los estudios sobre NLP son realizados por informáticos, aunque también hay interés por parte de lingüistas, psicólogos y filósofos. Las tareas investigadas en NLP incluyen la realización de resúmenes automática, la resolución de co-referencias, el análisis del discurso, la traducción automática, la segmentación morfológica, el reconocimiento de entidades nombradas, el reconocimiento óptico de caracteres y el etiquetado de partes del discurso, entre otras (Khurana, Koli, Khatte, et al., 2023).

Su objetivo es ayudar a los usuarios que no están familiarizados con los lenguajes específicos de las máquinas, facilitando la comunicación con computadoras en lenguaje natural. NLP es una rama de la inteligencia artificial y la lingüística, y se divide en dos partes principales: la comprensión del lenguaje natural (NLU) y la generación del lenguaje natural (NLG). NLU se centra en entender el lenguaje humano, mientras que NLG se ocupa de crear frases y párrafos significativos a partir de representaciones internas. En este trabajo nos centraremos solamente en el NLU (Natural Language Understanding).

Los primeros esfuerzos en NLP se basaban en reglas y gramáticas definidas a mano, con enfoques rule-based que dependían de diccionarios y reglas gramaticales. Estos sistemas eran rígidos y difíciles de adaptar a nuevas tareas o lenguajes.

La llegada del aprendizaje automático en los años 90 marcó un cambio, con algoritmos como Naïve Bayes y Support Vector Machines (SVM), que permitían a las máquinas aprender patrones en datos textuales. El uso de grandes volúmenes de datos y el entrenamiento supervisado mejoró el rendimiento de los sistemas, reduciendo la necesidad de reglas predefinidas.

En la última década, las redes neuronales profundas, como los modelos basados en transformers (BERT, GPT), han revolucionado NLP, aunque parece ser que el modelo BERT domina el mercado (Sawicki, Ganzha & Paprzycki, 2023). Estos modelos aprenden de manera más profunda, capturando contextos complejos y permitiendo una comprensión más sofisticada del lenguaje. Modelos como GPT-4 o BERT permiten tareas como la generación de texto, traducción automática, y más.

Las últimas innovaciones en NLP incluyen:

- **Large Language Models (LLM)** como GPT y BERT, que pueden generar texto, responder preguntas y realizar tareas complejas con gran precisión. Estos modelos se entrenan con vastas cantidades de datos y capturan el contexto mejor que las técnicas anteriores.
- **Transfer Learning:** Modelos preentrenados en grandes corpus de datos pueden ser afinados para tareas específicas, lo que reduce la necesidad de grandes volúmenes de datos para cada tarea nueva.

- **Inferencia Multimodal:** Combina texto con otras formas de datos, como imágenes o audio, para mejorar las aplicaciones de IA en dominios más complejos.

NLP se utiliza ampliamente en la vida diaria y en el ámbito empresarial. Algunas aplicaciones clave incluyen:

- Asistentes virtuales (como Siri o Alexa).
- Chatbots y servicio al cliente automatizado.
- Análisis de sentimientos en redes sociales.
- Traducción automática (Google Translate).
- Búsquedas semánticas y sistemas de recomendación.

Una de las ventajas de los modelos de NLP es que presentan mayor precisión en tareas como la traducción y el análisis de sentimientos. Además, NLP es esencial para automatizar y escalar procesos que implican lenguaje.

No obstante, los modelos más avanzados requieren grandes cantidades de datos y poder computacional. También existe el riesgo de sesgos en los modelos que pueden afectar la equidad de las decisiones automatizadas.

A pesar de los avances, NLP enfrenta varios desafíos:

- **Comprensión profunda del contexto:** Aunque los modelos actuales son potentes, no siempre comprenden el verdadero significado de las palabras, especialmente en contextos ambiguos. No comprenden las ironías, los sarcasmos ni los dobles sentidos.
- **Sesgos inherentes:** Los modelos pueden perpetuar sesgos de los datos con los que se entrenan, lo que plantea problemas éticos en aplicaciones como la contratación o la justicia.
- **Recursos computacionales:** El entrenamiento de modelos grandes requiere una gran cantidad de datos y poder computacional, lo que puede ser costoso y limitante.
- Se pueden consultar los progresos del NLP en tiempo real en la página web dedicada exclusivamente a ello: <https://nlpprogress.com/>

## <SENTIMENT ANALYSIS>

El análisis de sentimientos es una rama del procesamiento del lenguaje natural (NLP) que se centra en identificar y extraer información subjetiva de textos. Su objetivo principal es determinar la actitud del autor hacia un tema, ya sea positiva, negativa o neutral. Los dos enfoques principales son el análisis de sentimiento basado en reglas y el automatizado.

A pesar de los avances, aún existen desafíos significativos en el análisis de sentimientos:

- **Ironía y Sarcasmo:** Capturar el tono adecuado en textos que emplean ironía o sarcasmo sigue siendo complicado.
- **Multimodalidad:** La integración de texto, imágenes y otros tipos de datos para un análisis más holístico es algo a lo que se aspira, pero no es nada fácil de poner en práctica.
- **Sentimientos Complejos:** Los sentimientos pueden ser mixtos o complejos, lo que dificulta su clasificación.



## Análisis de sentimiento basado en reglas

La forma tradicional de hacer análisis de sentimiento se basa en un conjunto de reglas creadas manualmente. Este enfoque incluye técnicas de NLP como léxicos (lexicon en inglés, listas de palabras, en este trabajo nos referiremos a ellos como lexicón o lexicones), stemming (raíces de palabras), tokenización y análisis sintáctico.

El funcionamiento es sencillo: se crean «léxicos» o listas de palabras positivas y negativas que se utilizarán posteriormente para describir el sentimiento. Por ejemplo, los léxicos positivos pueden incluir «gracias», «perfecto» y «amigo». Los léxicos negativos podrían incluir «matar», «droga» y «enfadado».

Antes de analizar un texto, hay que prepararlo. Se utilizan varios procesos para formatear el texto de manera que una máquina pueda entenderlo.

### Tokenización:

- La tokenización divide el texto en pequeños fragmentos llamados tokens.
- La tokenización de frases divide el texto en frases.
- La tokenización de palabras separa las palabras de una frase. Por ejemplo, «el mejor servicio de atención al cliente» se dividiría en «el», «mejor» y «servicio de atención al cliente».

### Lemmatización:

- La lematización sirve para transformar las palabras en su raíz.
- Un lema es la forma raíz de una palabra. Por ejemplo, la forma raíz del verbo to be (ser, en inglés) «is, are, am, were y been» es «be».

### Exclusión de palabras no relevantes para el análisis de sentimientos:

- Se excluyen cosas que son conocidas pero que no son útiles para el análisis de sentimientos, como por ejemplo las coletillas (uh, eh, hmm, etc.)


### Eliminación de palabras vacías (stopwords y fillerwords):

- Se eliminan palabras comunes como «for, at, a, to» ya que estas palabras tienen poco o ningún valor semántico en la frase.

Seguidamente, un script realizado en Python cuenta el número de palabras positivas o negativas de un texto determinado. Una regla especial puede asegurarse de que las palabras negadas, por ejemplo «no es fácil», se cuenten como opuestas, aunque en este trabajo no hemos podido llegar tan lejos debido a la falta de tiempo.

El último paso consiste en calcular la puntuación general del sentimiento del texto. Como ya se ha mencionado, puede basarse en una escala de -1 a 1, aunque depende del lexicón en concreto. En este caso, una puntuación de 1 sería la máxima puntuación posible para un sentimiento positivo, una puntuación de 0 indicaría un sentimiento neutro y una puntuación de -1 indicaría un sentimiento negativo. Este tipo de análisis se conoce como **Análisis de Polaridad**.

Otra manera de contabilizar los sentimientos consiste en sumar un punto de un sentimiento concreto cada vez que se localiza una palabra asociada a él. Por ejemplo, si se encuentra la palabra “dead” (muerto) se suma un punto en la escala de sentimiento negativo, “happy” (contento, feliz) suma un punto en la escala de alegría y “tired” (cansado) suma un punto tanto en la escala de tristeza como en la de sentimiento negativo. Este tipo de análisis se denomina **Análisis de Sentimiento**.



Es importante tener en cuenta que este tipo de enfoque basado en reglas es limitado ya que no tiene en cuenta la frase en su conjunto. La complejidad del lenguaje humano hace que sea fácil pasar por alto negaciones y metáforas complejas, ironías y sarcasmos. Es por ello que desde los años 90 se intentan desarrollar mecanismos más sofisticados que reemplacen la tarea manual.

## **Análisis automatizado de sentimientos**

El análisis automatizado de sentimientos se basa en técnicas de aprendizaje automático. En este caso, se entrena un algoritmo de ML para clasificar el sentimiento basándose tanto en las palabras como en su orden. El éxito de este método depende de la calidad del conjunto de datos de entrenamiento y del algoritmo.

- **Paso 1: Extracción de características**

Antes de que el modelo pueda clasificar el texto, hay que prepararlo mediante tokenización, lematización y eliminación de palabras vacías al igual que en los enfoques basados en reglas. El algoritmo obtiene representaciones numéricas denominadas «características». Una forma habitual de hacer esto es utilizar los métodos de «bolsa de palabras». Estos métodos vectorizan el texto en función del número de veces que aparecen las palabras (también llamadas embeddings).

- **Paso 2: Entrenamiento**

En la siguiente fase, el algoritmo recibe un conjunto de datos de entrenamiento etiquetados con sentimientos. El modelo aprende a asociar los datos de entrada con la etiqueta correspondiente más adecuada. Por ejemplo, estos datos de entrada incluirían pares de características (o representaciones numéricas del texto) y su correspondiente etiqueta positiva, negativa o neutra. Los datos de entrenamiento pueden crearse manualmente o generarse a partir de las propias reseñas.

- **Paso 3: Predicciones**

En la etapa final es donde el análisis de sentimiento ML tiene la mayor ventaja sobre los enfoques basados en reglas. Se introduce un nuevo texto en el modelo. A continuación, el modelo predice etiquetas (también llamadas clases o etiquetas) para estos datos no vistos utilizando el modelo aprendido de los datos de entrenamiento. De este modo, los datos pueden etiquetarse como positivos, negativos o neutros. Esto elimina la necesidad de un léxico predefinido que se utiliza en el análisis de sentimientos basado en reglas.

## **Algoritmos de clasificación**

Los algoritmos de clasificación se utilizan para predecir el sentimiento de un texto concreto. Se entrenan utilizando datos de entrenamiento preetiquetados. Los modelos de clasificación suelen utilizar Naive Bayes, Logistic Regression, Support Vector Machines, Linear Regression y Deep Learning.

**Naive Bayes:** este tipo de clasificación se basa en el Teorema de Bayes. Se trata de algoritmos probabilísticos, es decir, calculan la probabilidad de una etiqueta para un texto concreto. A continuación, el texto se etiqueta con la etiqueta de mayor probabilidad. Las palabras individuales contribuyen de forma independiente y equitativa al resultado global. Esta suposición puede ayudar a que este algoritmo funcione bien incluso cuando los datos son limitados o están mal etiquetados.



## Algoritmos de Deep Learning

Nos habría gustado poder explorar modelos de deep learning como el Long Short-Term Memory o LSTM, las Redes Neuronales Recurrentes (RNN), las Redes Neuronales Convolucionales (CNN) y los Transformers y Modelos Preentrenados (como BERT, GPT) pero por limitaciones en la capacidad de cómputo no lo hemos podido realizar. Lo que sí hemos hecho es utilizar transformers junto con el modelo RoBERTa para realizar una clasificación zero-shot de las frases del guión de la serie, aunque sin finetunearlo ni preentrenarlo con ningún texto.

## 3. DESARROLLO\_

Antes de comenzar a trabajar, hemos decidido crear un Google Drive compartido por motivos de practicidad y para que el código pueda ser consultado por todos los integrantes del equipo en todo momento. Este Drive no es el creado por Immune, ya que parece haber problemas de permisos con este último. También hemos creado un grupo de Discord compartido en el que participan todos los integrantes del equipo y el tutor.

### 1. Lectura y aprendizaje teórico de lo que es el NLP y el Sentiment Analysis.

En esta primera etapa nos hemos enfocado en entender qué es el NLP y en qué consiste su subsección: el análisis de sentimientos. También hemos buscado en Github repositorios dedicados a este tipo de proyectos para ganar know-how. Por último, hemos analizado la manera en la que han sido realizados para ser capaces de organizar nuestras ideas y establecer nuestro punto de partida y un roadmap de trabajo.

### 2. Creación de un primer excel con datos básicos: Temporada, Episodio, Personajes y Textos del guión de la serie.

El primer punto de nuestro roadmap consiste en generar un dataset con la información básica de la serie: decidimos que lo único que necesitamos son el número de temporada, el número de episodio, el título de cada capítulo, el nombre del personaje que habla en cada momento y sus frases, es decir, el guión de la serie.

Generamos un Excel con una pestaña por capítulo, decidimos no hacer un csv para no tener una lista única y gigante de texto y poder explorar y analizar con mayor comodidad los textos de cada episodio. Más tarde nos encargamos, utilizando Python, de unir todas las pestañas para crear un dataframe unificado.

Comenzamos nuestra búsqueda de transcrips (guiones) y los encontramos en la página web <https://transcripts.foreverdreaming.org/viewforum.php?f=165>, no obstante, nos damos cuenta de que los nombres de los personajes no figuran en los guiones de las temporadas 4 y 5. Este es un problema estratégico que debemos solucionar, ya que, si no podemos separar los diálogos de Walter de los del resto de personajes, no podremos analizar su descenso a la maldad.

Valoramos tres posibles técnicas para conseguir obtener los nombres de los personajes que nos faltan:

- a) Visionar la serie y anotar manualmente los nombres de los personajes al lado de sus frases
- b) Utilizar técnicas de Deep Learning para, mediante el análisis de los audios de la serie, poder extraer los diálogos, y, acto seguido, también mediante técnicas de Deep Learning, poder asociar cada voz a cada speaker.
- c) Intentar mediante prompt engineering que la IA Generativa (ChatGPT) nos aporte esta información (no lo conseguimos porque los guiones de Breaking Bad están protegidos con Copyright).

Tras fallar el intento con ChatGPT, y con el fin de ser más eficientes y completar esta tarea lo antes posible, intentamos seguir por la vía del Deep Learning. Más concretamente, intentamos las siguientes herramientas:

- i. **PYANNOTE:** Pyannote es una biblioteca de Python utilizada en tareas de análisis de voz. Está diseñada para realizar una técnica conocida como “diarización del habla” (identificación de quién habla y cuándo). Para aplicar esta técnica, extraemos el audio en formato .wav del video del primer capítulo de la serie Breaking Bad mediante una aplicación online. No obstante, al lanzar los scripts, nos damos cuenta de que es extremadamente lento en nuestro ordenador, necesita aproximadamente 1,5 horas para terminar un capítulo. Además, sólo detecta los speakers en un 55% de las frases, el resto los clasifica como “unknown”, con la dificultad añadida de tener que, posteriormente, identificar cada speaker (mediante visionado de la serie) y reemplazar por su nombre el valor atribuido Pyannote y de tener que verificar los speakers con frases vacías y de los ruidos de fondo (observando la imagen se entiende mejor el problema).

1181	Speaker SPEAKER_15:
1185	Speaker SPEAKER_15: Grab his shoe.
1189	Speaker SPEAKER_15: Let us work.
1194	Speaker SPEAKER_15: Oh. Here. Wait, wait.
1195	Speaker SPEAKER_15: We're
1196	Speaker SPEAKER_15: We're ready to go to work.
1200	Speaker SPEAKER_15: You kill Jesse,
1204	Speaker SPEAKER_16: I don't know. Pick one.
1208	Speaker SPEAKER_16: You can't afford to do this.
1212	Speaker SPEAKER_16: You won't do this.
1217	Speaker SPEAKER_16: You're too smart.
1222	Speaker SPEAKER_17: [footsteps]
1226	Speaker SPEAKER_17: Do you have your keys?
1227	Speaker SPEAKER_17: on the right.
1228	Speaker SPEAKER_17: Right as rain.
1229	Speaker SPEAKER_17: Uh, it's parked
1230	Speaker SPEAKER_17: Yeah.
1234	Speaker SPEAKER_17: Yeah.
1235	Speaker SPEAKER_18: Closed!

- ii. **WHISPER Y SPLEETER:** Intentamos también combinar Whisper con Spleeter, ya que son complementarios: Spleeter divide el audio en voz y ruido de fondo (música, ruido ambiental, etc.) y whisper extrae el texto de las voces. Lo que queremos conseguir con esto es mejorar la identificación de los speakers por parte del Pyannote, no obstante, tampoco funciona adecuadamente.
- iii. **TRANSFORMERS:** Finalmente, y como último recurso, decidimos entrenar un

modelo con las primeras tres temporadas e intentamos que prediga los personajes de las temporadas 4 y 5. Lo tenemos que parar porque tarda demasiado y tampoco vemos resultados.

Tras mucho buscar en internet en español, inglés y francés, conseguimos encontrar las transcripciones de algunos capítulos con los nombres de los personajes que hablan de las temporadas 4 y 5 en esta página web: <https://genius.com/Breaking-bad-cornered-script-annotated>

Decidimos pasar a la siguiente etapa con el transcript completo, pero con los personajes únicamente presentes en las primeras 3 temporadas y en los capítulos 6 y 11 de la temporada 4 y 7, 9, 13, 14, 15 y 16 de la temporada 5.

En un segundo tiempo, al darnos cuenta de que necesitamos más datos de Walter White de las temporadas 4 y 5, volvemos a esta etapa y extraemos los nombres de los personajes de los capítulos 1, 2, 3, 4, 5, 6, 8, 10, 11 y 12 de la temporada 4 (la temporada 4 queda, pues, completa), y de los capítulos 1, 2 y 3 de la temporada 5 (ésta no queda completa por falta de tiempo para seguir con la extracción).

### 3. Preparación del dataset inicial

Comenzamos la fase de preparación de los datos disponibles mediante el formateo de un Excel que contendrá los datos iniciales sobre los que basaremos nuestro Sentiment Analysis. Este Excel se llama **transcripts.xlsx**.

Es muy importante que todas las pestañas tengan la misma estructura para que Python sea capaz de separar la información correctamente. Se decide una estructura en la que la primera fila contendrá el número de la temporada, el número del episodio y el título del episodio. A partir de la segunda fila encontraremos el guión con la siguiente estructura: personaje:frase (sin espacios).

El procedimiento detallado es el siguiente:

- Se selecciona el texto en la página web que lo contiene y se pegan los valores en Excel. Todo se pega en la columna A. Seleccionamos esta columna y la convertimos en tabla (en el menú superior *Insertar* → *Tabla*).
- Acto seguido se añade un filtro a esta columna A que contiene los datos y se filtra por las celdas vacías. Seleccionamos estas filas vacías pinchando en ellas desde la izquierda (donde están los números de fila) y las eliminamos. Quitamos el filtro para comprobar que todas nuestras filas con texto están ahora seguidas, sin filas vacías entre medias.
- Tras esta verificación, eliminamos el filtro completamente pinchando en *Datos* → *Filtro* en el menú superior y quitamos el formato tabla yendo a *Diseño de Tabla* → *Convertir en Rango*. Ahora disponemos del texto en formato plano, sin espacios en blanco entre fila y fila y con nuestro formato deseado **personaje:frase** en las primeras tres temporadas.
- No obstante, antes de seguir nos damos cuenta de que debemos formatear los capítulos de las temporadas 4 y 5 que no tienen nombres de personaje de la misma manera. Repetimos la operación para eliminar filas vacías entre el texto.

Introducimos una nueva columna A (nuestro texto queda en la columna B) y la rellenamos con ":". Una vez hecho esto, utilizamos la función Concatenar de Excel para unir los dos puntos con el texto. Copiamos y pegamos como valores la nueva columna para retirar la fórmula y que el json parser no falle. Eliminamos el resto de las columnas de forma que nuestra nueva columna concatenada quede en la columna A. Su formato final será **vacío:frase**.

- Se limpian todas las comillas simples y dobles que pudieran interferir con el json parser, se limpian también los asteriscos que figuran en las palabras malsonantes y se reemplazan por las palabras enteras (por ejemplo, sh\*t se reemplaza por shit, f\*ck por fuck, etc.). Es muy importante que no haya ninguna tabla, ni encabezado, ni filtro en ninguna de las pestañas, ya que, si los hay, el json parser fallará.

Este procedimiento de limpieza se realiza en todas las pestañas del Excel (una pestaña por cada uno de los 62 episodios). Una vez que está todo limpio y el formato es homogéneo, lo guardamos y pasamos a la siguiente etapa.

#### 4. Importación de primeras librerías y limpieza básica del dataset:

Esta etapa se divide en dos subetapas:

##### a) Montar drive e importar librerías

Generamos el primer documento de Google Colab destinado a trabajar toda la parte de preprocesamiento. Empezamos a importar las librerías de base que no pueden faltar en ningún proyecto:

- **Pandas:** biblioteca de Python para manipulación y análisis de datos. Proporciona estructuras de datos como DataFrames y Series
- **Matplotlib:** biblioteca de visualización de datos en Python. Se utiliza para crear gráficos simples y complejos.
- **Seaborn:** biblioteca basada en Matplotlib, diseñada para hacer visualizaciones estadísticas más atractivas y fáciles de interpretar.

También añadimos dos librerías necesarias para el manejo de archivos de gran tamaño como es nuestro caso:

- **Gdown:** biblioteca de Python que se utiliza para descargar archivos desde Google Drive utilizando su URL compartida. Es particularmente útil para descargar archivos grandes o para automatizar el proceso de descarga desde Google Drive.
- **Openpyxl:** biblioteca de Python que permite leer y escribir archivos de Excel con el formato .xlsx.

Estas dos últimas librerías son absolutamente necesarias para garantizar que la importación de datos en Excel se realiza de manera correcta. A pesar de que es más común importar los datos en formato csv, hemos querido conservar el formato Excel por dos motivos: porque es más fácil manejar un dataset con 62 pestañas en Excel que un csv larguísimo y porque mantener el formato .xlsx nos permite exportar nuestro trabajo también en este formato de manera sencilla y directa. Esto facilita las verificaciones tanto del dataset original como del del trabajo realizado.

Creamos el script de carga del fichero Excel y realizamos las transformaciones necesarias

para generar el dataframe que necesitamos. En este paso, es muy importante añadir al `read_excel` el `index_col=None` y `header=None`, ya que si no pandas añade un índice (algo que no necesitamos) e interpreta la primera fila como header (algo que tampoco queremos puesto que vamos a crear nosotros los nombres de las columnas que necesitamos y, además, la primera fila contiene información que queremos replicar en sus respectivas columnas).

Ya que nuestro Excel es muy grande, está en formato Excel, y dispone de 62 pestañas, nos ha parecido necesario implementar una estructura `try-except` con el fin de detectar y subsanar con mayor facilidad los errores que pudieran surgir en el momento de la carga.

Como output de este script obtenemos otro archivo Excel, llamado `final_df`, que tiene el formato que necesitamos y que utilizaremos en el posterior análisis de sentimientos y clustering. Este dataframe cuenta con las cinco variables siguientes: Temporada, Episodio, Título, Personaje y Texto.

### **b) Limpieza básica del dataframe `final_df`**

Ya tenemos nuestro dataframe **`final_df`**, pero todavía no está listo para ser utilizado. Tenemos que realizar una limpieza básica del dataset.

Empezamos por homogeneizar los nombres de los personajes principales para asegurarnos que, más adelante, cuando tengamos que analizarlos, no perdamos parte de sus diálogos.

No nos fijamos en los personajes secundarios, ya que hay muchos nombres desconocidos debido a la extracción manual de datos de las temporadas 4 y 5. Esto no es relevante porque suprimimos todos los personajes que tienen menos de 340 líneas de diálogo, lo cual nos reduce el dataset a 16402 líneas. Nos acordamos de volver a descargar el **`final_df`** con menos filas que el inicial.

Al eliminar los personajes secundarios, nos quedamos sólo con los personajes principales. Nos parece útil crear una nueva variable que se llame `Personaje_id` que nos permita filtrar por personaje más adelante si es necesario y de forma que se los podamos pasar a los algoritmos de Machine Learning (ya que estos sólo aceptan variables numéricas).

Para crear esta nueva variable `Personaje_id`, nos apoyamos en la librería `Labelencoder` de `Skicit-Learn` especializada la conversión de etiquetas categóricas en valores numéricos. Esto nos evita el trabajo de tener que dar un número a cada personaje manualmente. Los personajes quedan numerados por orden alfabético y nuestro protagonista Walter White pasa a ser el personaje número 36.

Por último, realizamos un check rápido del tipo de cada variable. Queremos que las variables `Temporada` y `Episodio` sean numéricas por el mismo motivo que el evocado anteriormente: poder pasárselas a los algoritmos de clustering más adelante. Hacemos la conversión.

## **5. Preprocesamiento**

Ahora sí estamos listos para comenzar la fase de preprocesamiento. Este apartado se divide en varias secciones.



### a) Importar y entender el funcionamiento de las librerías de extracción de tokens.

El objetivo de este apartado es convertir los datos estructurados (las frases) en datos estructurados (palabras con significado). Para ello, vamos a utilizar dos bibliotecas muy importantes en NLP:

- **Biblioteca “re”:** se utiliza para trabajar con expresiones regulares. Las expresiones regulares (o regex) son patrones que se utilizan para buscar, coincidir o manipular cadenas de texto. Con re, puedes realizar tareas como buscar patrones dentro de textos, validar formatos, extraer información y realizar reemplazos.
- **Biblioteca “nltk” (Natural Language Toolkit):** proporciona una amplia gama de funcionalidades para analizar, manipular y extraer información de los datos textuales, entre otras, la tokenización, la lematización y el stemming, el etiquetado del discurso (POS tagging), el análisis de la estructura gramatical, el reconocimiento de las entidades nombradas (NER), y la extracción de palabras clave entre otros.

De estas dos bibliotecas vamos a importar los siguientes recursos:

- **from nltk.corpus import stopwords:** importamos el recurso stopwords del módulo corpus perteneciente a la biblioteca nltk. Las stopwords son palabras que habitualmente se filtran en el procesamiento de texto debido a su alta frecuencia y bajo significado semántico.
- **from nltk.tokenize import word\_tokenize:** importamos el recurso word\_tokenize del módulo tokenize de nltk. word\_tokenize es una función específica dentro de nltk que se utiliza para dividir un texto en tokens que son palabras. Esta función toma una cadena de texto como entrada y devuelve una lista de palabras, separando los signos de puntuación de lo que son las palabras propiamente dichas.
- **from nltk.stem import WordNetLemmatizer:** stem se refiere al stemming, proceso de reducción de palabras a su forma base o raíz. En cuanto al WordNetLemmatizer, es una herramienta de lematización que utiliza la base de datos de WordNet (<https://wordnet.princeton.edu/>) para reducir las palabras a su forma base, conocida como lema.
- **nltk.download('punkt'):** Es el tokenizador de NLTK, el algoritmo que lleva a cabo la tokenización de las palabras. Divide el texto en frases a través de un modelo no supervisado, esto es que no necesita datos etiquetados para su entrenamiento, obtendrá patrones de los propios datos.
- **nltk.download('stopwords'):** recurso que contiene una lista de palabras “irrelevantes” en varios idiomas. Hace que no haya que escribirlas una a una manualmente para que el preprocesamiento no las tenga en cuenta.
- **nltk.download('wordnet'):** wordnet es una base de datos léxica del idioma inglés que agrupa palabras en conjuntos de sinónimos (synsets) y proporciona información semántica (significados, antónimos, relaciones entre palabras).
- **Fillerwords:** al realizar un chequeo de las stopwords, nos damos cuenta de que contiene prácticamente todas las palabras que necesitamos quitar, por lo tanto, no consideramos necesario definir una lista de “filler words” (palabras de relleno) a suprimir.



En este punto, tras inicializar el WordLemmatizer, convertir en colección (set) las “stop words” del idioma inglés y asegurarnos de que no tenemos ningún valor nan (not a number, celda vacía) en la variable que vamos a preprocesar (que recordemos que era la variable Texto, la que contenía todos los diálogos de la serie), damos comienzo al preprocesamiento.

Definimos la función que preprocesa el texto, aprovechando también para convertir todo el contenido de la variable Texto a minúsculas. Esta función realiza las siguientes acciones en este orden:

- Aplica la librería “re” (regex) que nos quita los caracteres especiales,
- Tokeniza las frases contenidas en cada celda de la variable texto y almacena las palabras extraídas en una variable llamada “words” (palabras),
- Lemmatiza una a una las palabras contenidas en la variable “words” creada por el tokenizador mediante una list comprehension y sobreescibe la variable “words” con las palabras lematizadas,
- Y retorna esta variable “words”.

Aplicamos esta función a nuestro final\_df, columna Texto, y almacenamos el resultado en una nueva variable llamada Palabras\_clave. Esta variable será la que utilizaremos para realizar nuestro Sentiment Analysis. Con la creación de esta columna de palabras clave, hemos conseguido transformar datos no estructurados (conversaciones entre personas) en datos estructurados (listas de palabras con significado y sentido propio a analizar). Con nuestro dataset en Excel completo con esta nueva columna, pasamos a la siguiente fase.

## 4. PRUEBAS Y RESULTADOS

Hemos realizado Análisis de Sentimientos desde las diferentes perspectivas mencionadas en el apartado State of the Art: Reglas, Machine Learning Supervisado y Machine Learning No Supervisado.

Los resultados obtenidos no son del todo satisfactorios, por diversos motivos que se analizarán en la sección de Conclusiones. Veamos, pues, cuáles han sido las pruebas realizadas y que se ha podido concluir en cada una de ellas.

### 1. Extracción de sentimientos – Rule-based approach

En esta parte rule-based, nos fijaremos especialmente en la evolución de la polaridad e intensidad de las emociones y sentimientos de Walter, tanto positivos, negativos como neutros. Algunos índices calculan un valor compound, o “promedio”, que también analizaremos si lo consideramos relevante.

#### <TEXTBLOB>

Textblob es una librería de procesamiento de texto que permite realizar tareas de NLP como análisis morfológico, extracción de entidades, análisis de opinión, traducción automática, etc.

Está construida sobre otras dos librerías muy famosas: NLTK y pattern. Su principal ventaja es que permite combinar el uso de las dos herramientas anteriores en un interfaz más simple.

<b>Escala:</b>	de -1 (polaridad estrictamente negativa) a 1 (polaridad estrictamente positiva), 0 significa neutral (no hay polaridad). Mide la Polaridad.
----------------	--

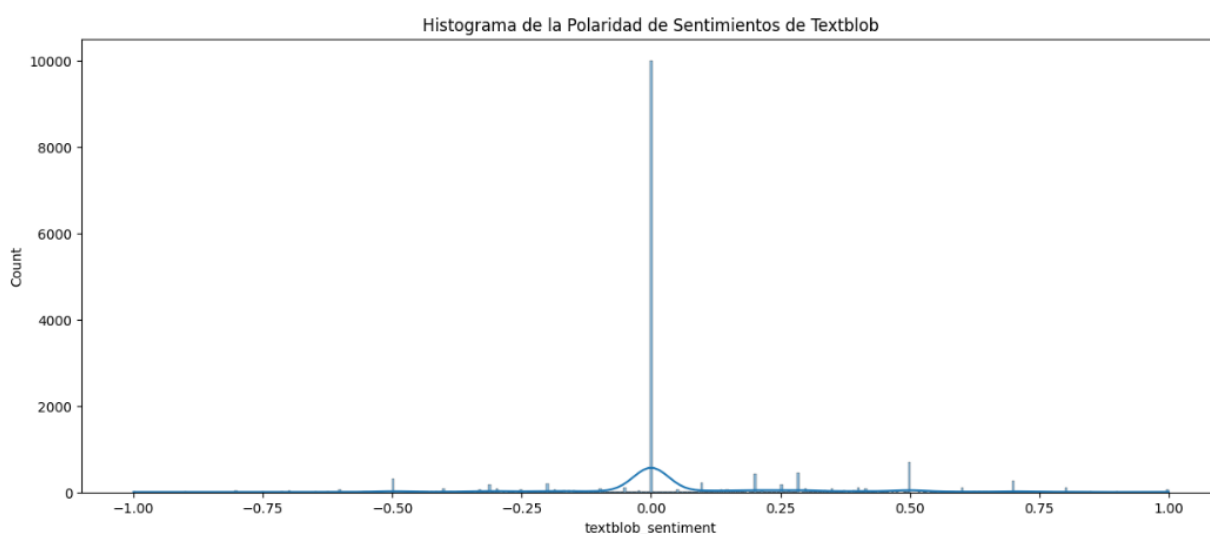
Definimos la función que extraerá la polaridad del sentimiento para cada fila de nuestra nueva variable Palabras\_clave, asegurándonos que no tenemos valores nulos en nuestra ella. También hacemos un pequeño check para limitar el número de decimales en caso de que nuestra variable contenga flotantes (en nuestro caso no los contiene, pero lo dejamos por si acaso).

Almacenamos el output en una nueva variable llamada "textblob\_sentiment". Podemos, opcionalmente, descargar el dataset final\_df para realizar un chequeo visual de nuestra nueva variable y realizar un primer análisis del scoring de polaridad atribuido por Textblob a cada frase. También realizamos una inspección en pantalla mediante el método .head() de python.

Al realizar este chequeo, nos damos cuenta de que muchas de las palabras más fuertes (matar, muerto, palabras malsonantes) son puntuadas de manera bastante "neutral" por Textblob, no puntúan lo suficientemente cerca del -1 como deberían. Es decir, percibimos un cierto "sesgo positivo" por parte de Textblob, probablemente porque no está afinado para captar lenguaje soez y malsonante ni para captar sarcasmo, ironía ni dobles sentidos.

En ese sentido, nos proponemos elaborar un lexicón custom donde alteremos manualmente las puntuaciones de las palabras negativas más extremas y ver si esto corrige el sesgo (lo veremos más adelante, en la siguiente sección).

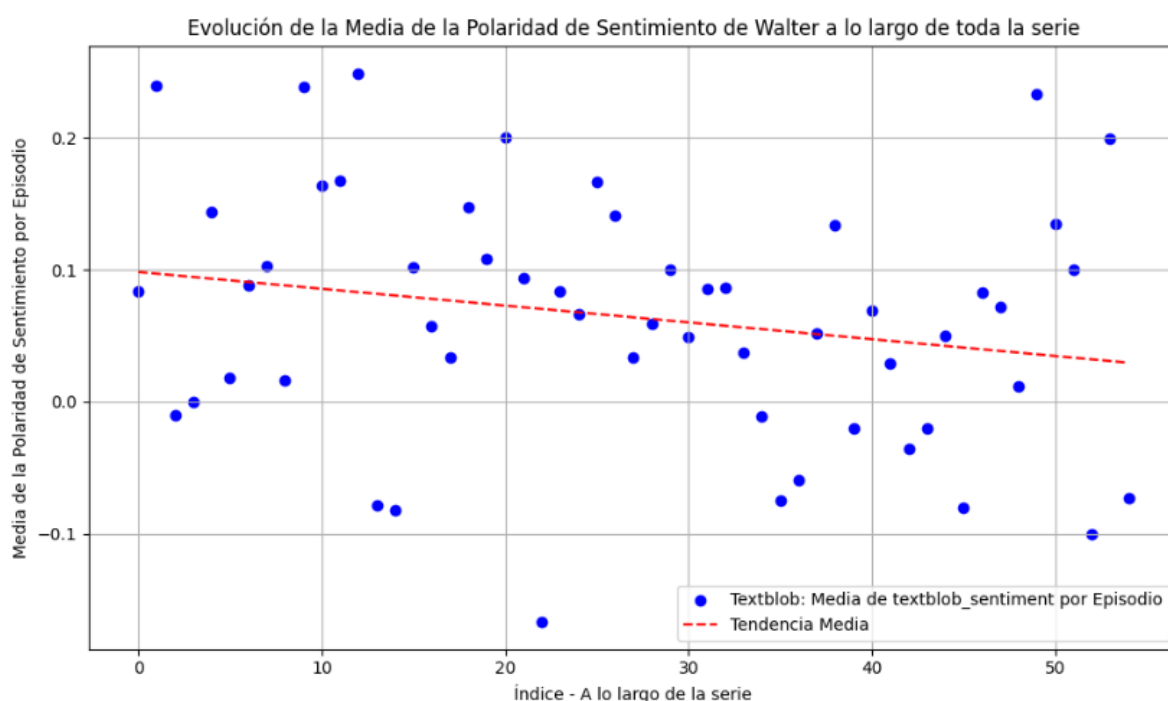
Seguimos analizando nuestros resultados del Textblob. Con el fin de tener una mejor idea de la distribución del scoring de Textblob, realizamos un histograma. La enorme cantidad de nulos llama instantáneamente nuestra atención:



Comprendemos que no podemos trabajar con tal cantidad de valores neutros, por ello, con el fin de equilibrar la distribución, eliminamos el 90% de los neutros comprendidos entre -0,2 y +0,2. De los 16402 valores neutros iniciales, pasamos a 1019.

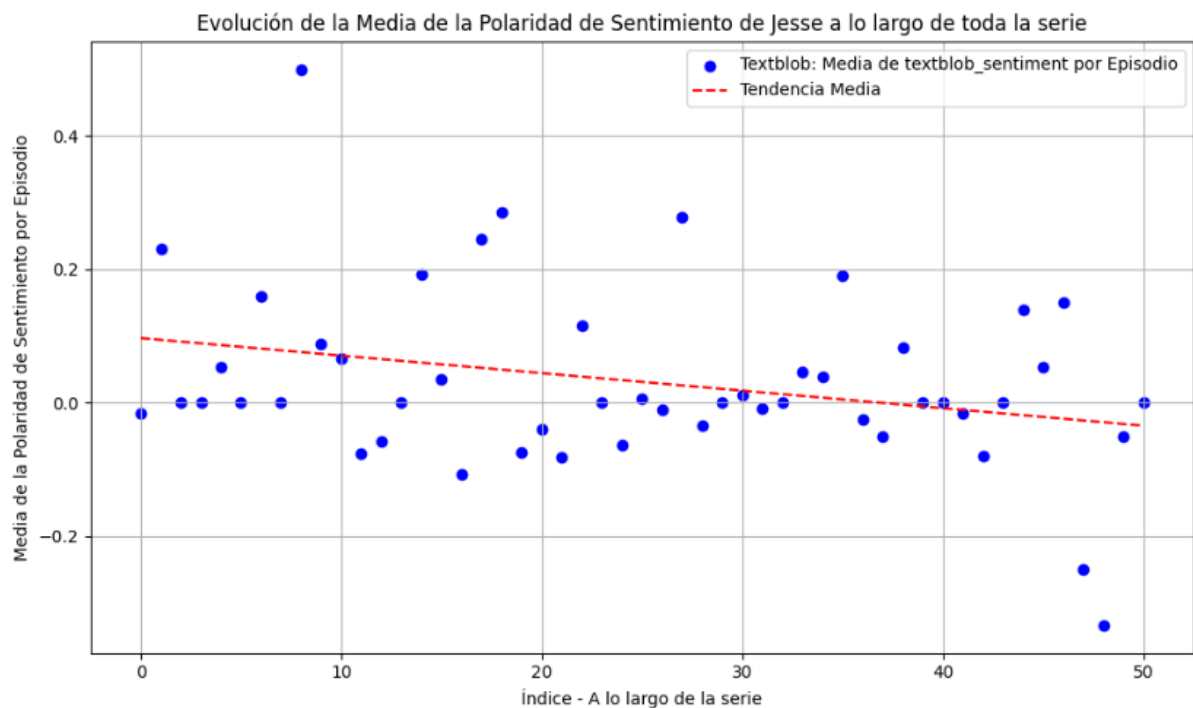
Al realizar este ajuste, nos encontramos con la primera representación del descenso a la maldad de Walter White al realizar la gráfica de la evolución de la media de la Polaridad de Sentimiento de Walter.

Un pequeño apunte antes de seguir avanzando: la elección de la media por encima de otras medidas para obtener la polaridad agregada responde a una cuestión de practicidad, estamos más acostumbrados a interpretar medias que, por ejemplo, medianas. Entendemos la ventaja, entre otras, de utilizar medianas en el manejo de outliers, pero ante la incapacidad de abarcar todas las métricas nos hemos decantado por la media en este trabajo.



Como hemos comentado previamente, si realizáramos un ajuste de los outliers (valores atípicos, es decir, puntos de datos que se encuentran significativamente alejados de los demás puntos), probablemente podríamos ajustar mejor la línea de tendencia. No obstante, preferimos dejarlo así ya que muestra con mayor veracidad los altibajos de este personaje a lo largo de la serie.

Para verificar que esto tiene sentido, comprobamos que el socio de Walter en la serie, Jesse, traficante de poca monta al principio y verdadero mafioso al final, también sufre esta metamorfosis hacia la maldad:



La distribución de Jesse es más clara que la de Walter, posiblemente porque no tiene relación con su familia y tiene muchas menos interacciones positivas con otras personas que Walter, lo cual contribuye a que sus puntuaciones negativas no se “reequilibren” con las positivas como sí ocurre en el caso de Walter. De hecho, vemos que el descenso a la maldad de Jesse es más acusado que el de Walter (Jesse baja de 0,1 a -0,04 aproximadamente y Walter de 0,1 a 0,2).

## <TEXTBLOB CON CUSTOM LEXICON DE PALABRAS NEGATIVAS>

Como hemos dicho anteriormente, nos parece interesante volver a realizar el Textblob pero añadiéndole una custom bad-word list (una lista de malas palabras, el Excel se llama **bad\_words\_for\_textblob.xlsx**) cuyos scores hayamos manipulado hacia la polaridad negativa para intentar sesgar el análisis hacia la maldad.

Este archivo se llama bad\_words\_for\_textblob.xlsx y contiene 622 palabras que aparecen en los guiones de la serie Breaking Bad junto con un score negativo que le hemos asignado nosotros a mano.

Este Textblob es idéntico al anterior (tiene la misma escala entre el -1 y el 1) salvo en que nuestro lexicón custom prevalecerá sobre el lexicón estándar de Textblob cuando la función encuentre una de las palabras que hemos incluido en nuestra lista.

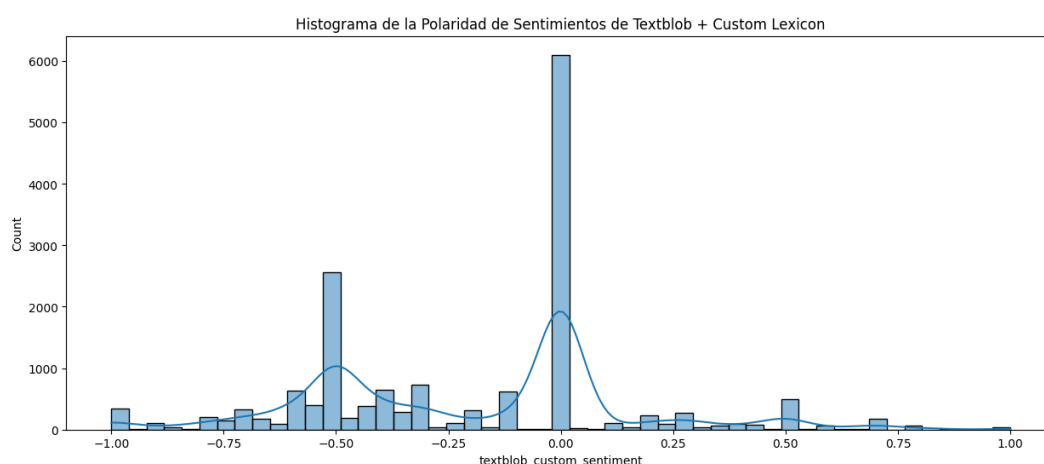
Por ejemplo, a la frase *“If you're gonna bring a gun, you gotta bring enough gun. 40 caliber.”*, el Textblob normal le atribuye una puntuación de 0, es decir, considera que esa frase es neutral. Sin embargo, Textblob con nuestro lexicón customizado atribuye a esta misma frase una polaridad de -0,5, captando mejor el sentido negativo de la misma.

No obstante, en otras ocasiones, es Textblob Custom el que “sobreinterpreta” algunas frases, como por ejemplo en el caso de esta frase: *“All right, come on, come on. All right. School bus is clear. Got the green light.”*. Textblob la puntúa con una polaridad ligeramente positiva mientras que Textblob Custom la puntúa con -0,5 puntos, lo cual es, a todas luces, incorrecto y una exageración.

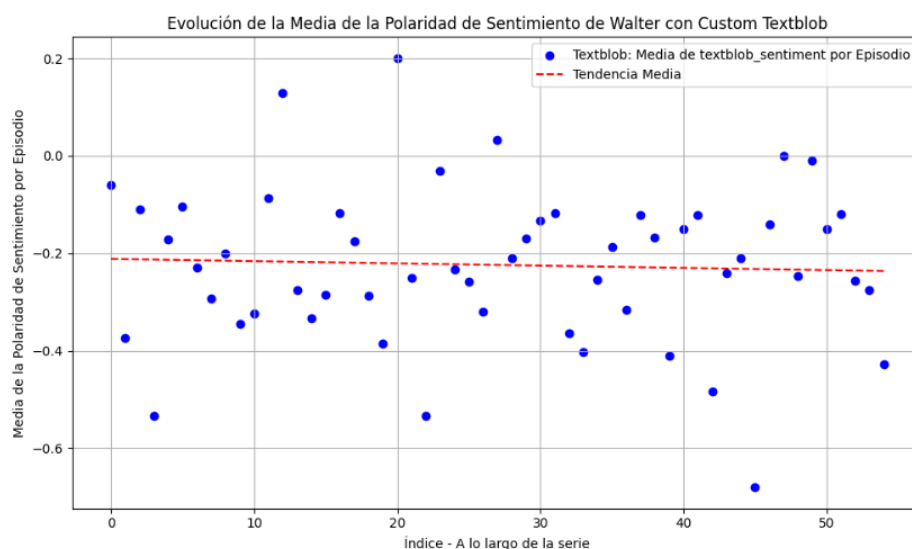
En un tercer caso, ninguno de los dos lexicones interpreta correctamente la frase. En la frase *“Don't worry about the baby. This is just for you. We are just doing you tonight. So just close your eyes. Relax, and let it. Close your eyes.”* que tiene una connotación claramente positiva, esta positividad no es captada ni por el Textblob normal ni por el Textblob Custom, ya que le atribuyen puntajes de polaridad de 0 y -0,55 respectivamente.

En el siguiente gráfico podemos observar el efecto que nuestra manipulación de los scores negativos ha tenido sobre la distribución, hay muchos menos valores neutros y se aprecia un “clúster” secundario en la parte negativa del eje x (hacia -0,5).

Además, ya que la mayoría de los datos se encuentran en la parte izquierda de la distribución, podemos decir que presenta una asimetría positiva (cola hacia la derecha):



El resultado es el siguiente:



La conclusión que podemos extraer de esta sección es que la lista customizada con scores manipulados no nos modifica sustancialmente la distribución, es más, su performance es peor que la del Textblob normal que hemos presentado en el apartado anterior para modelar el descenso de Walter a la maldad.

Lo único que podemos destacar es que hemos conseguido que casi todas las medias caigan por debajo de la línea del 0, es decir, hemos conseguido realizar una traslación vertical, o hacia el lado negativo, de las medias de los datos originales.

Quizás deberíamos haber incluido en nuestra lista palabras positivas con scorings positivos manipulados. Otra alternativa habría sido utilizar librerías ya existentes, como por ejemplo el profanity-check, pero hemos tenido que descartar la idea por falta de tiempo.

Por todo ello, no vemos el coste-beneficio de modificar manualmente los scorings negativos y no pensamos que sea la solución para mostrar adecuadamente el descenso de Walter a la maldad.

### <VADER (Valence Aware Dictionary and sEntiment Reasoner)>

Vader, es un lexicón diseñado específicamente para textos de redes sociales. Funciona bien con textos cortos e informales, aunque no detecta bien el sarcasmo ni la ironía. Cabe destacar que Vader tampoco entiende bien las negaciones. Por ejemplo, "not bad" sería clasificado como malo por Vader.

No obstante, teóricamente sí que toma en cuenta los signos de puntuación y las mayúsculas, así como las palabras colindantes que pudieran modificar el significado del conjunto.

Por este motivo, hemos decidido no pasarle a Vader la variable Palabras\_clave, que recordemos que era el resultado del preprocesamiento de la variable Texto, sino la variable Texto directamente.

Por aclararlo aún más, hemos hecho que Vader no analice palabras sueltas sino las frases enteras tal cual figuran en el guión.

Con todo esto nos hacemos una idea de las ventajas y limitaciones de este lexicón, y, a pesar de que su uso esté recomendado para análisis de sentimientos de tweets o comentarios en redes sociales, decidimos probarlo porque nuestro guión también son datos no estructurados.

<b>Escala:</b>	<p>Cuatro valores: positivo, negativo, neutro, compound.</p> <ul style="list-style-type: none"><li>• <b>NEUTRAL, POSITIVO o NEGATIVO: <math>0 &lt; x &lt; 1</math></b> (los valores se ubican en columnas distintas, pero son siempre positivos entre 0 y 1)</li><li>• <b>COMPOUND:</b> de -1 a 1. Es una puntuación de sentimiento generalizado y se calcula con una fórmula específica:</li></ul> <div><math display="block">\text{compound} = \frac{\sum_{i=1}^n S_i}{\sqrt{(\sum_{i=1}^n S_i^2) + \alpha}}</math></div> <ul style="list-style-type: none"><li>• <math>S_i</math> es el puntaje de cada palabra individual.</li><li>• <math>n</math> es el número total de palabras que tienen un puntaje.</li></ul>
----------------	---

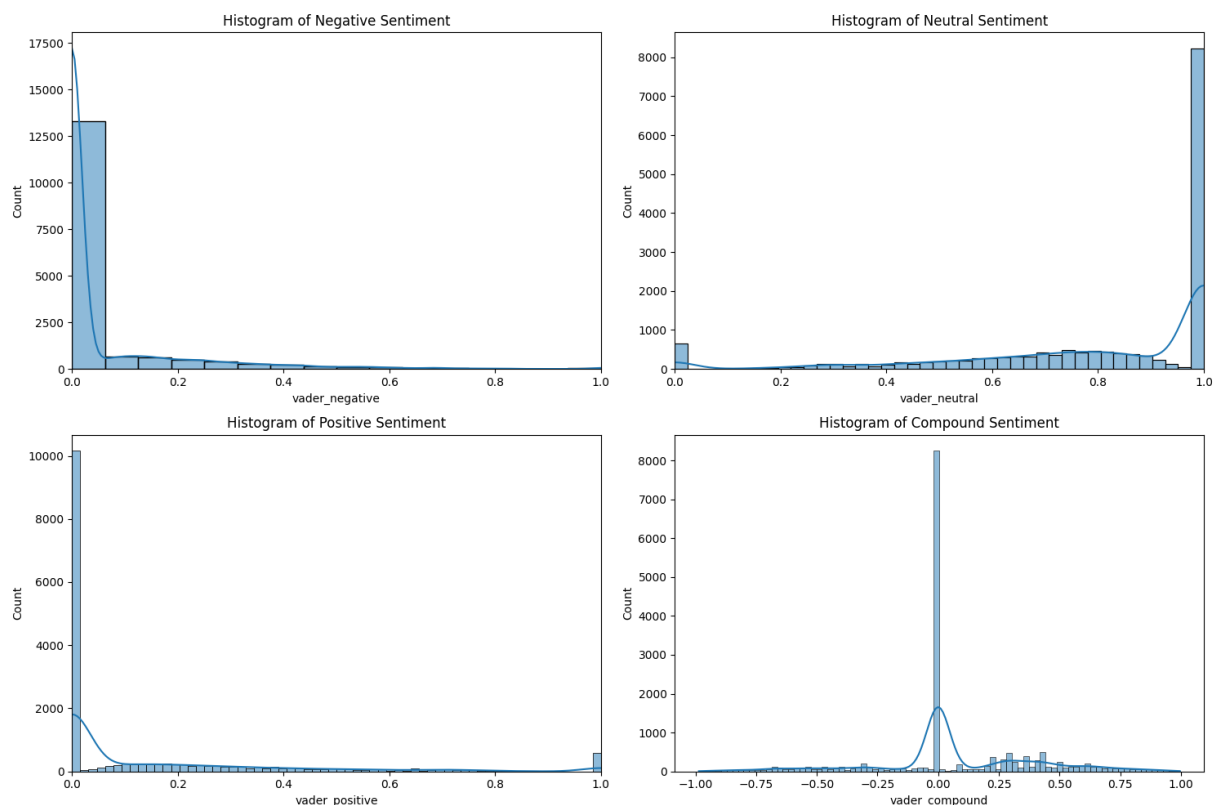


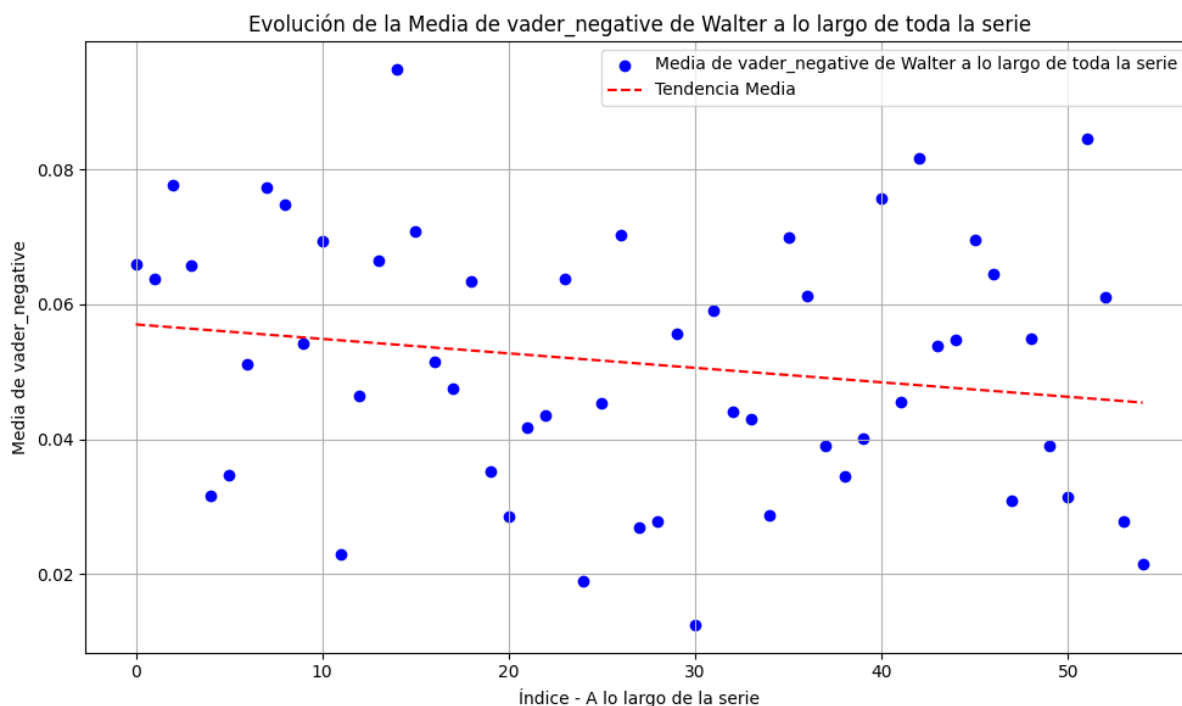
- $\alpha$  es un pequeño valor de ajuste (normalmente 15) que estabiliza el resultado.

Mide el **Sentimiento** en el caso del **Positivo, Negativo y Neutral**, y la **Polaridad** en el caso del **Compound**.

Sacamos los histogramas y vemos que VADER clasifica la mayoría del texto muy cerca del 0, es decir, no es capaz de comprender las emociones fuertes, sobre todo las negativas.

También vemos que hay una mayoría de sentimiento negativo. No obstante, al hacer los scatterplots y analizar la línea de regresión de Walter, esta mayoría de negativos es descendente a medida que avanza el tiempo. Es decir, Walter sería cada vez menos negativo con respecto al principio de la serie. Por el momento no le encontramos explicación a esto, aunque pensamos que este resultado debe deberse a las limitaciones del modelo a la hora de entender el sarcasmo y la ironía más presentes cuanto más avanza la serie.





## <SENTIWORDNET>

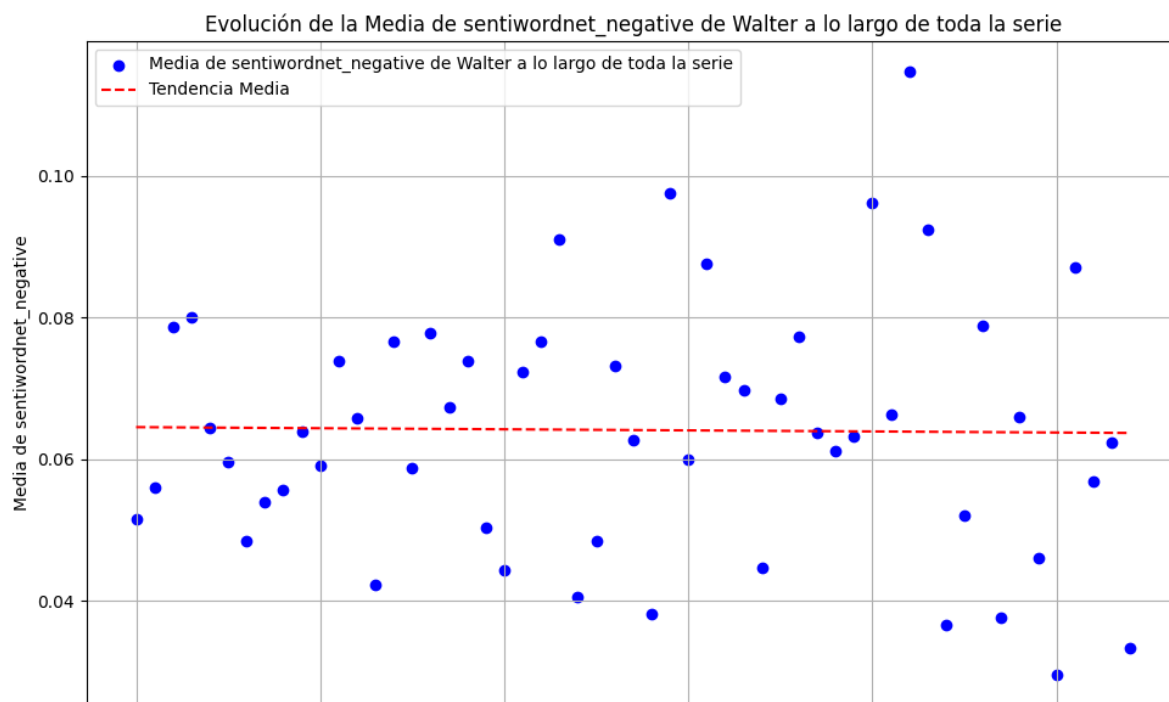
Sentiwordnet tampoco es un lexicón adaptado al contenido que estamos intentando analizar en este trabajo, ya que está diseñado para analizar opiniones en lo que se conoce como Opinion Mining y no está adaptado para analizar diálogos entre personas. No obstante, decidimos probarlo por pura curiosidad.

Algunos usos de este lexicon incluyen el análisis y comprensión de las opiniones de los consumidores sobre determinados productos o el estudio de las opiniones de los votantes sobre los distintos candidatos a la presidencia de un país.

A partir de la base de datos WordNet, este léxico asigna puntuaciones a los synsets (grupos de palabras con el mismo significado) según crea que se está emitiendo una opinión positiva, negativa o neutra.

<b>Escala:</b>	<b>Positivity score (PosScore):</b> indica cuan positiva es una frase de 0 a 1. <b>Negativity score (NegScore):</b> indica cuan negativa es una frase de 0 a 1. <b>Objectivity score:</b> se calcula implícitamente realizando la operación siguiente $\rightarrow 1 - (\text{PosScore} + \text{NegScore})$
----------------	---

No nos vamos a detener demasiado en este Lexicón ya que, tal y como esperábamos, no funciona bien. Detecta el mismo nivel de negatividad en Walter al principio y al final de la serie, aunque es cierto que los puntos están más dispersos hacia el final que al principio:



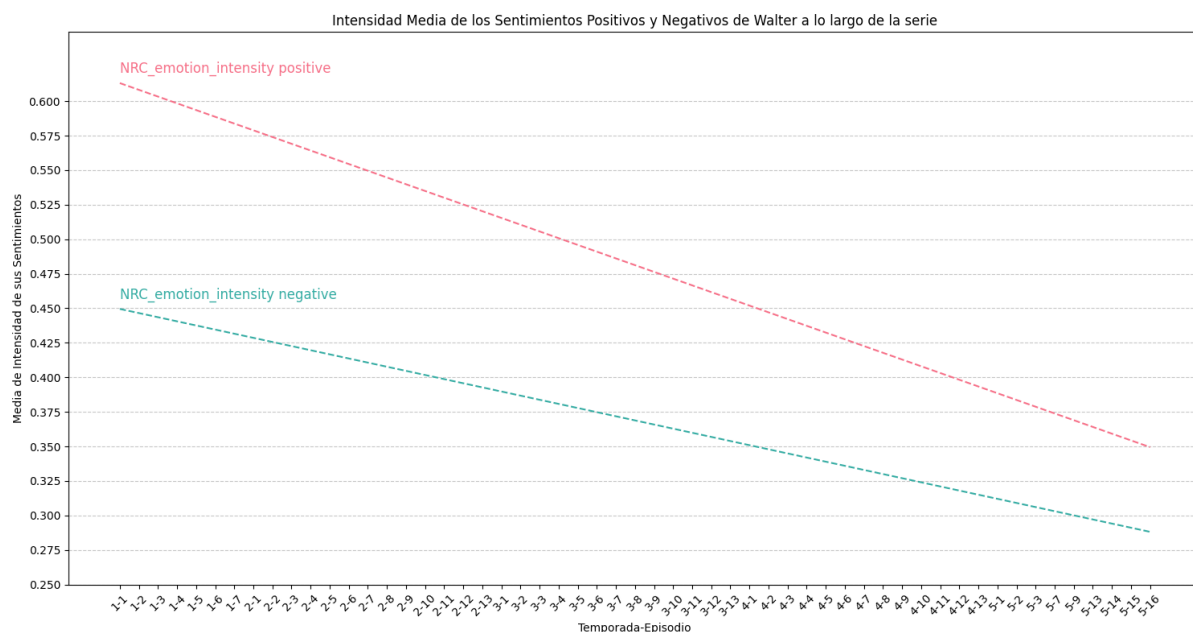
## <NRC EMOTION LEXICON>

El Léxico de Emociones del NRC es una lista de palabras inglesas y sus asociaciones con ocho emociones básicas (ira, miedo, anticipación, confianza, sorpresa, tristeza, alegría y asco) y dos sentimientos (negativo y positivo) desarrollado por el National Research Council de Canadá.

Cada palabra de este lexicon puede pertenecer a una categoría o a varias. Por ejemplo, la palabra abandono puntuará en las emociones tristeza y miedo y en el sentimiento negativo.

<b>Escala:</b>	<p>Suma un punto a las categorías a las que pertenece una palabra cada vez que la encuentra sin límite máximo ni mínimo.</p> <p><b>Sentimientos: 2</b> -&gt; negativo y positivo</p> <p><b>Emociones: 8</b> -&gt; ira, miedo, anticipación, confianza, sorpresa, tristeza, alegría y asco.</p> <p>Mide el <b>Sentimiento</b></p>
----------------	--

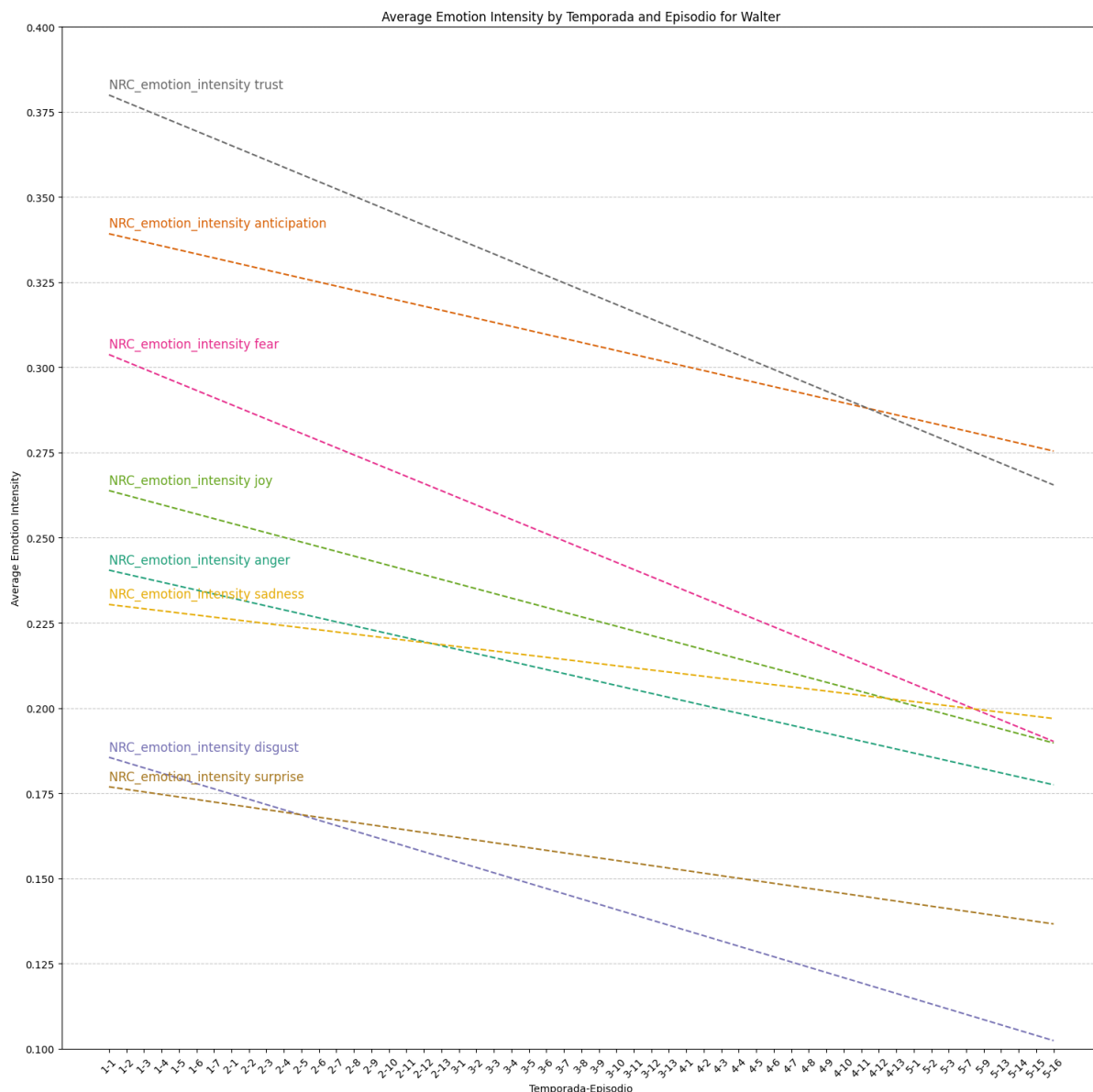
Nos interesamos por los dos sentimientos en un primer momento. Al instante de realizar el gráfico nos percatamos de que Walter presenta una atenuación emocional muy importante, ya que tanto sus sentimientos positivos como sus sentimientos negativos se ven muy reducidos al final de la serie con respecto al principio de la serie:



Es decir, Walter es notablemente más frío al final de la serie que al principio, su capacidad de sentir sentimientos positivos se encuentra notablemente reducida, más aún que su capacidad de sentir sentimientos negativos, que también se encuentra muy alterada.

Según nos indica este lexicón, su desensibilización emocional es patente y siente una mayor indiferencia emocional hacia sus acciones y las consecuencias que estas tienen en sí mismo y en su entorno más cercano.

Pasamos a analizar las 8 emociones. Al realizar el gráfico, el panorama se repite: Walter muestra menores niveles de todas las emociones medidas por el lexicón NRC.



Por el lado de las emociones positivas, a lo largo de los episodios Walter ha perdido, en gran parte, la alegría, la capacidad de sorprenderse, la capacidad de confiar y la capacidad de anticipar. Por el lado de las emociones negativas, Walter ha perdido la capacidad de disgustarse, de sentir tristeza, de sentir miedo y de sentir enfado.

Al final de la serie Walter siente mucho menos de todo tipo de emoción. Claramente se ha deshumanizado.

## <ZERO-SHOT CON EL MODELO RoBERTa Y LA LIBRERÍA TRANSFORMERS DE HUGGING FACE>

Para terminar con nuestra sección de preprocesamiento, intentamos una aproximación un poco distinta.

Decidimos realizar un zero-shot con el modelo de lenguaje preentrenado RoBERTa (A Robustly Optimized BERT Pretraining Approach) desarrollado por Facebook AI en 2019.

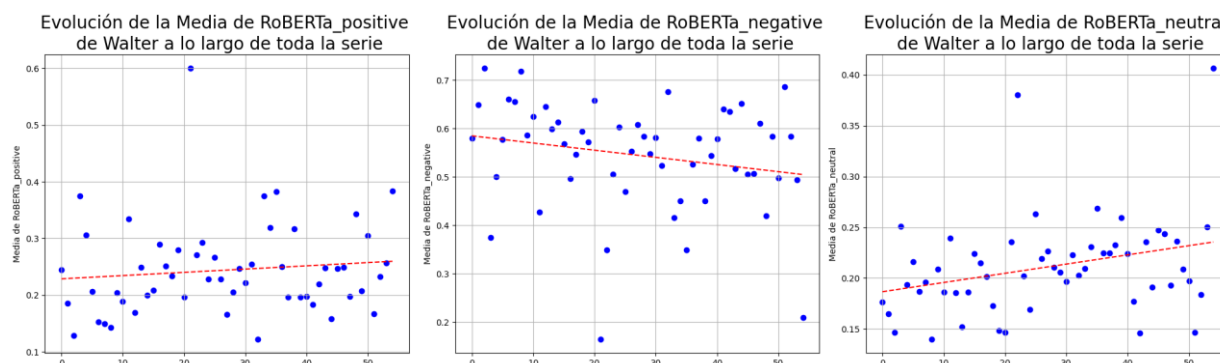
Este modelo se basa en una arquitectura de Transformers, específicamente en una variante de BERT (Bidirectional Encoder Representations from Transformers) mejorada. Ha sido diseñado para realizar tareas de procesamiento del lenguaje natural (NLP) con una alta precisión ya que fue entrenado con 160GB de texto sin procesar, aunque necesita ser fine-tuneado para alcanzar su funcionamiento óptimo.

A pesar de no contar con los recursos ni el tiempo necesarios para realizar el finetuning, hemos decidido hacer un zero-shot (técnica de machine learning que permite a un modelo clasificar clases que no ha visto durante su fase de entrenamiento) porque creemos que su aportación puede sernos de ayuda.

Nota: Ya que el script tarda entre 4 y 6 horas en ejecutarse en un ordenador normal, facilito el archivo Excel con los datos: **RoBERTa\_classification.xlsx**.

<b>Escala:</b>	Adjudica a cada frase un puntaje <b>positivo, negativo y neutro</b> . Estos tres puntajes siempre suman <b>1</b> . Es decir -> <b>positivo + negativo + neutro = 1</b> en todos los casos.
----------------	---

La ejecución del script tarda varias horas (entre 4 y 6 dependiendo de la CPU disponible). Realizamos la regresión lineal de los scatterplots de las medias de los capítulos para Walter y observamos el siguiente gráfico:



En un principio, creemos obtener un resultado contrario a lo esperado, pero mirando detenidamente los tres gráficos nos damos cuenta de que son totalmente coherentes con lo comentado en el punto anterior. Estos gráficos representan la deshumanización que sufre Walter a lo largo de la serie:

- Los sentimientos positivos apenas aumentan, y, verificando las frases, podemos ver que provienen del exceso de confianza que va desarrollando a la largo de su carrera criminal.
- Los sentimientos negativos disminuyen significativamente, lo que es coherente con lo visto en los gráficos del NRC: ya no siente miedo, tristeza ni asco con lo que está haciendo.
- El aumento significativo de neutros también valida los resultados del NRC: su tono cada vez es menos exaltado, menos extremo, más frío, y, por tanto, más neutro.



Pequeña nota al margen para terminar: somos conscientes de la presencia de outliers en esta distribución, pero decidimos dejarlos para que se pueda apreciar mejor que no todos los episodios tienen la misma carga emocional y que de un episodio al siguiente el tono de un personaje puede cambiar radicalmente.

Terminamos así la parte de preprocesamiento, pero no nos vamos con las manos vacías. Al aplicar los distintos lexicones hemos ido alimentando nuestro `final_df` con nuevos datos, y, al final de esta sección, hemos conseguido construir un nuevo dataset, llamado **ready\_df**, que nos servirá para profundizar en el análisis.

## 2. Machine Learning Supervisado

### <NAIVE BAYES>

El algoritmo Naive Bayes es un enfoque probabilístico que realiza predicciones basadas en pruebas y probabilidades previas. Se utiliza a menudo en tareas de clasificación de textos porque es sencillo y eficaz. En nuestro caso, lo utilizaremos para predecir si un texto es positivo o negativo y ver si, con sus predicciones, conseguimos modelar el descenso de Walter hacia la maldad.

Este algoritmo se basa en el concepto de probabilidad condicional. Supone que la presencia de una característica concreta en una clase no está relacionada con la presencia de otras características en esa clase. Este supuesto se denomina independencia condicional de clase.

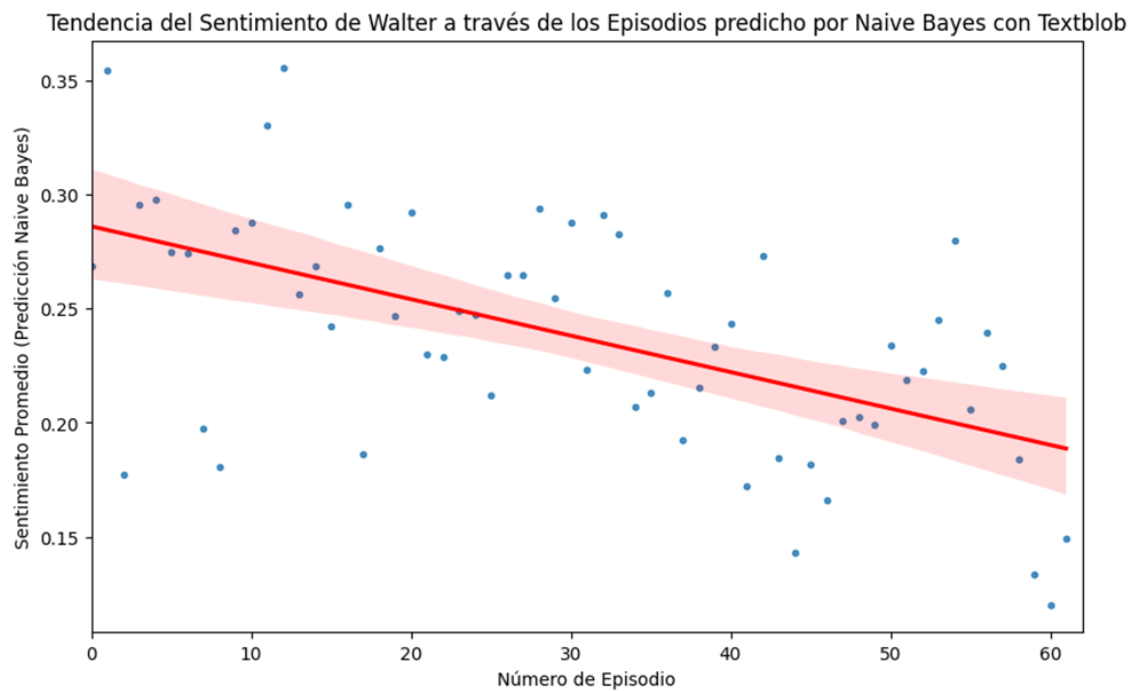
Puede aplicarse a cualquier problema de clasificación en el que tengamos un conjunto de características y queramos predecir la etiqueta de clase.

En nuestro caso, utilizamos esta técnica para ver si consigue predecir el descenso a la maldad de Walter. Para ello dividimos nuestro `ready_df` en dos sets: `train` y `test`. El set de entrenamiento cuenta con el 70% de los datos y el de prueba con el otro 30%. Vectorizamos ambas variables y se las pasamos al modelo Multinomial Naïve Bayes.

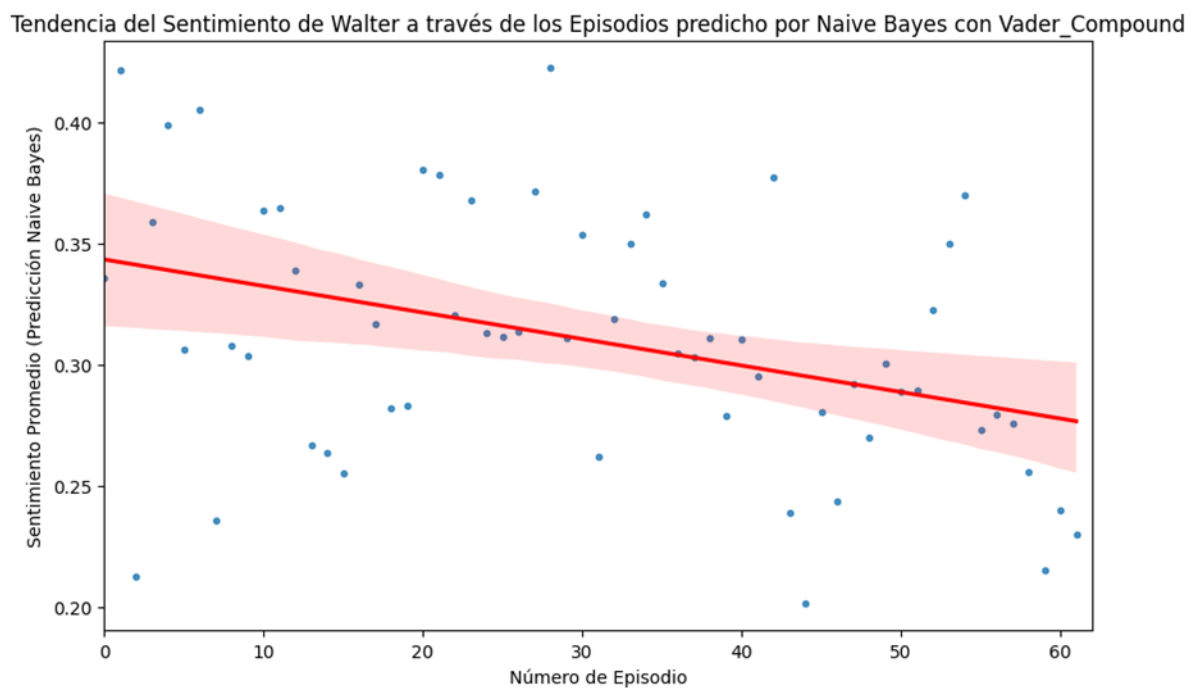
La variable que le pasamos al algoritmo es la de Texto, es decir, las frases completas, ya que antes de aplicar el algoritmo hemos realizado una vectorización y hemos aprovechado para eliminar las stopwords.

Al realizar los gráficos vemos que predice adecuadamente (con buena accuracy) el descenso de Walter a la maldad, por lo tanto, si en un futuro quisiéramos utilizar este algoritmo para modelar predecir los scores de los episodios que faltan o para modelar cualquier otra serie, sabríamos que podemos fiarnos de él y no tendríamos que repetir con los nuevos datos todo el trabajo previamente realizado.

Con la variable **Textblob** obtenemos una **accuracy del 91%**:

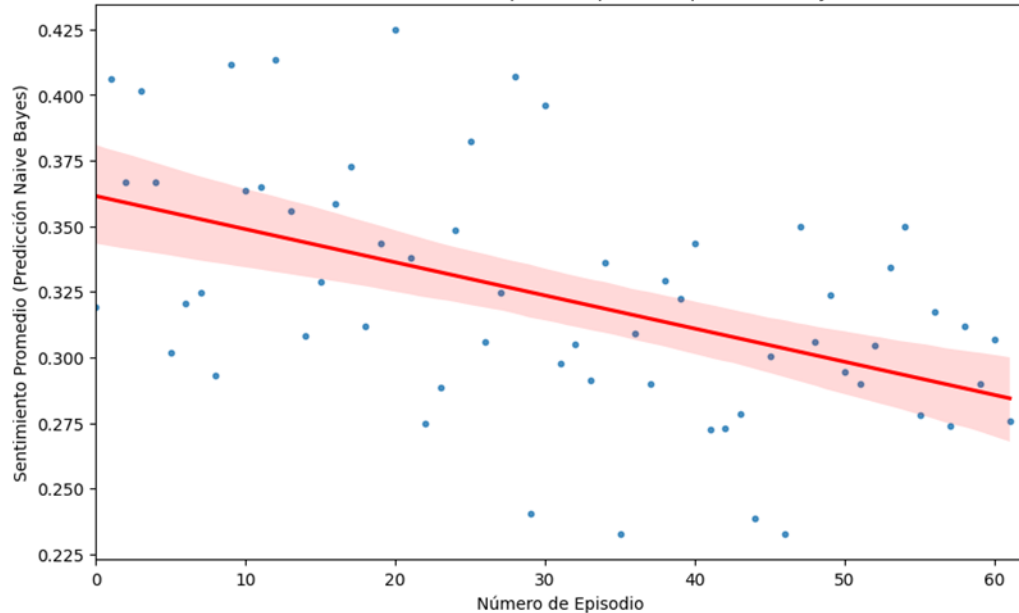


Con la variable **Vader\_compound** obtenemos una **accuracy del 89%**:



Y con la variable **Sentiwordnet\_compound** una accuracy del 86%:

Tendencia del Sentimiento de Walter a través de los Episodios predicho por Naive Bayes con Sentiwordnet\_compound

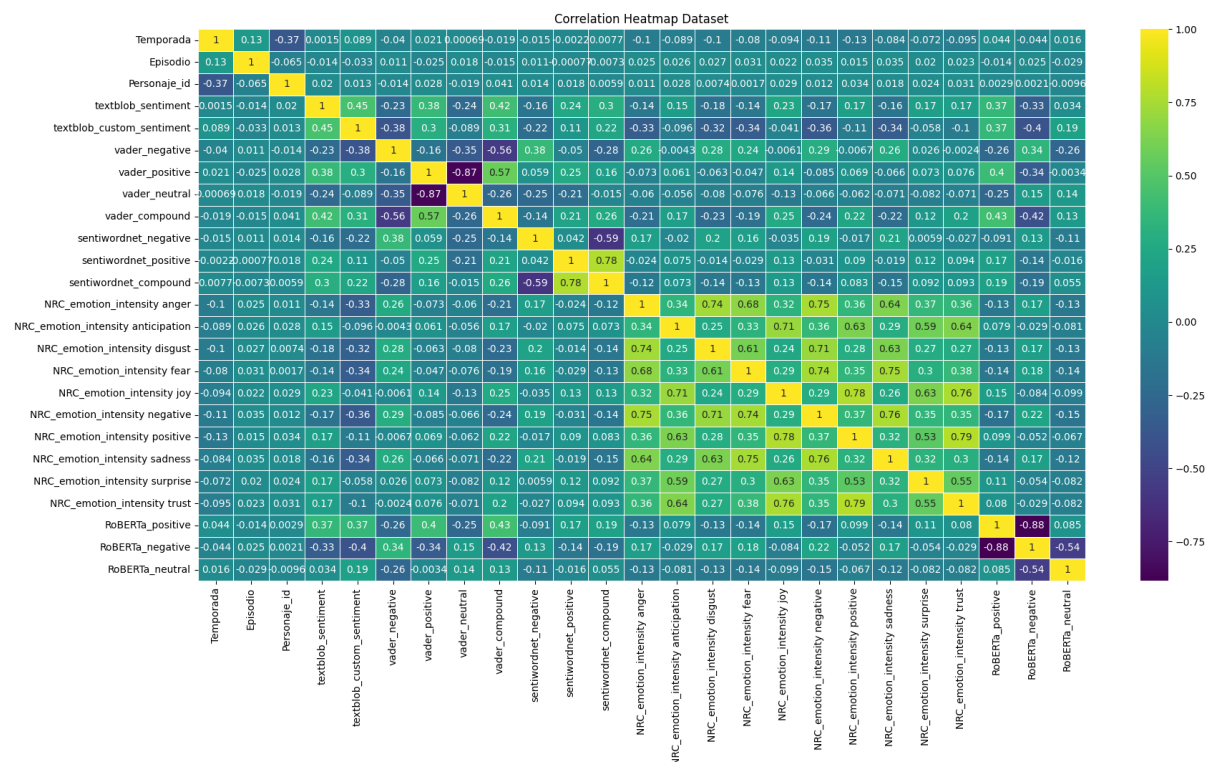


Por lo tanto, Naïve Bayes consigue predecir adecuadamente la progresión descendente del Walter y puede ser reutilizado en futuras ocasiones.

### 3. Machine Learning No-Supervisado

Con nuestro dataframe ready\_df bajo el brazo, nos disponemos a ponerlo bajo la lupa del Machine Learning No-Supervisado.

#### <MATRIZ DE CORRELACIONES ENTRE LOS DISTINTOS LEXICONES>



Con el fin de hacernos una idea de lo que tenemos en ready\_df, lo primero que hacemos es cargar el dataset, quitamos al personaje “scene” ya que al no ser un personaje real no lo vamos a analizar y sacamos una matriz de correlaciones de Pearson. En esta matriz podemos visualizar la relación entre las variables que ya sabíamos que están relacionadas (las que pertenecen al mismo Lexicón) pero también las relaciones entre las variables que, a priori, no tienen nada que ver entre sí.

En este sentido, en las correlaciones entre variables que no pertenecen al mismo conjunto, nos fijamos en las correlaciones positivas moderadas del textblob con el textblob custom (0,45) y del textblob con el Vader\_compound (0,42). Tanto el textblob como el Textblob custom muestran una correlación baja pero significativa con el RoBERTa\_positive (0,37) y una correlación negativa baja-moderada con el RoBERTa\_negative (-0,33 y -0,4 respectivamente).

También es llamativo que el NRC esté “captando” en cierto sentido nuestra manipulación de las escalas negativas del TextBlob Custom, mostrando correlaciones negativas de entre -0,32 y -0,36.

Resulta interesante, asimismo, que Vader\_negative correlacione con el RoBERTa\_negative (0,34), mientras que le Vader\_positive y el Vader\_compound correlacionan igualmente con el RoBERTa\_positive (0,4 y 0,43 respectivamente). Por tanto, no ocurre lo mismo con los neutros, entre los que no se aprecia ninguna correlación destacable.

A pesar de todo esto, el lexicón que parece sobresalir por encima de los demás es el NRC, con correlaciones altas entre las emociones negativas, y correlaciones moderadas y altas también entre las emociones positivas (desde 0,55 hasta 0,79). Las correlaciones entre emociones negativas y positivas son más bajas pero significativas, lo que da como resultado un modelo robusto para medir los altibajos emocionales de nuestro protagonista.

Por último, el lexicón que peor parece correlacionar peor con los demás es el Sentiwordnet, cuyos resultados, recordemos, tampoco nos han convencido a nosotros.

## <CLUSTERING CON K-MEANS>

Empezamos con el clustering seleccionando el algoritmo K-Means, que, a pesar de no ser mejor adaptado a un dataset tan complejo como el que nos traemos entre manos, hemos pensado que nos podría aportar insights interesantes.

K-means es un algoritmo de clasificación no supervisada que agrupa objetos en  $k$  grupos o clústers basándose en sus características. Estos clústers se construyen para contener puntos de datos «similares» entre sí y «disímiles» de los puntos de datos asignados a otros clústers. El agrupamiento se realiza minimizando la suma de distancias entre cada objeto y el centroide de su grupo o clúster. Para ello utiliza la distancia euclidiana, que es la distancia “normal”, es decir, la distancia más corta que conecta dos puntos.

Con nuestro `ready_df` cargado, nos aseguramos de ponerlo a punto para que el K-Means lo pueda tratar. Eso significa eliminar las variables categóricas. Creamos un nuevo `df` llamado `X` que contiene sólo las variables numéricas y con el cual operaremos de aquí en adelante.

Inspeccionando nuestro dataframe `X`, nos damos cuenta de que, aunque todas las variables son numéricas, no todas son continuas. Tenemos varias variables discretas, 3 en concreto: Temporada, Episodio y Personaje Id. Este problema, de sobra conocido por los científicos de datos, se conoce como **Mixed Data Clustering**.

El Mixed Data Clustering es un problema porque la distancia euclidiana, que es la que usa K-Means, no es apropiada para variables categóricas ni numéricas discretas. Su utilización cuando no se debe puede interferir en la correcta creación de los clústers, realizando agrupaciones falsas o sin sentido y haciendo que se complique la interpretación de los resultados.

Existen varias soluciones que pasan por utilizar escalas especiales para escalar estos datos discretos o por el uso de técnicas de codificación. Las escalas alternativas más conocidas son la Distancia de Gower, la Distancia Manhattan o la Distancia de Hamming, y la técnica de codificación más habitual es la conocida como one-hot-encoder.

En nuestro caso, y por simplificar el problema, decidimos escalar solamente las variables numéricas continuas, extrayendo las discretas del dataset temporalmente hasta que puedan ser reincorporadas al dataset escalado.

Decidimos normalizar las variables continuas con el Standard Scaler, también llamado Z-Score Scaler. El Standard Scaler escala las características de los datos de tal manera que tengan una media de 0 y una desviación estándar de 1. Es sensible a los outliers por lo que funciona mejor cuando la distribución es normal (o casi normal).

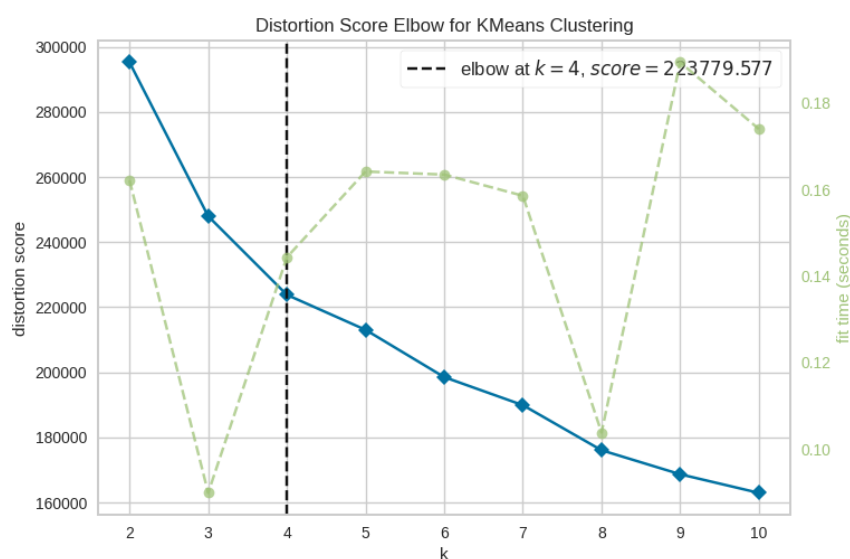
Nos gustaría realizar un pequeño inciso en este punto para explicar la diferencia entre escalar y normalizar. La normalización es un tipo específico de escalado que consiste, en el caso de la normalización Z-Score, en transformar los datos para que estos adquieran ciertas propiedades estadísticas, como una media de 0 y una desviación estándar de 1. Existen otros tipos de escalado que no son “normalizados”, es decir, hacen que los valores caigan en rangos específicos a definir en cada caso.

Los datos normalizados quedan, pues, de la siguiente manera:

	textblob_sentiment	textblob_custom_sentiment	vader_negative	vader_positive	vader_neutral	vader_compound	sentiwordnet_negative
0	0.294791	-0.554050	-0.180957	-0.118162	0.203223	2.406718	-0.045781
2	-0.239147	0.565786	-0.403959	-0.589298	0.762002	-0.237633	-0.478618

2 rows x 8 columns

Una vez hecho esto, creamos nuestro Diagrama del Codo. Esta herramienta nos ayuda a saber cuántos clústeres debemos pedirle que cree a K-Means, ya que este algoritmo requiere que le pasemos este parámetro de antemano.



Vemos que nos recomienda crear 4 clústeres (línea vertical rayada negra), por lo que aplicamos ese parámetro k a nuestro algoritmo K-Means.

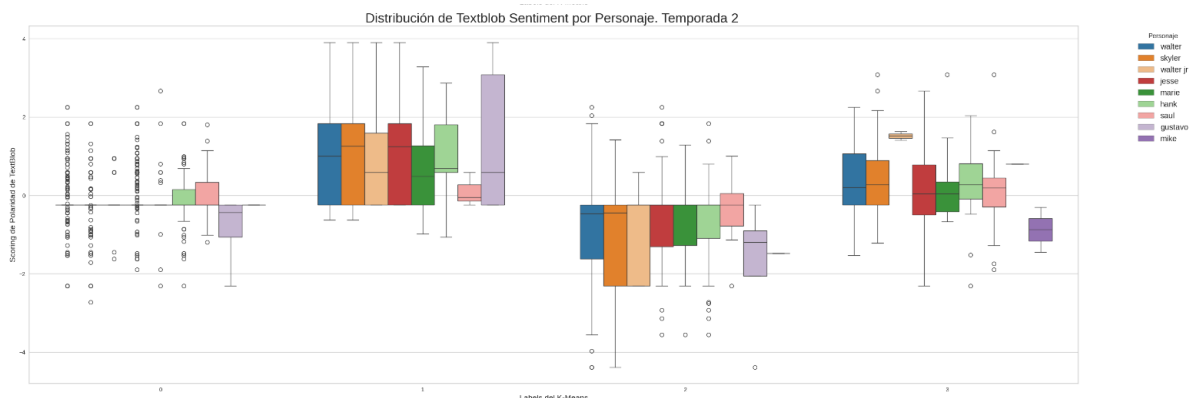
Una vez creados nuestros 4 clústeres, damos comienzo a su análisis. Como tenemos que analizar el descenso a la maldad de Walter White, vamos a realizar un análisis por temporada, es decir, vamos a ver qué ocurre con Walter en cada clúster en cada temporada.

Comenzamos por el Textblob. No vamos a analizar el Sentiwordnet, ya que no nos fijamos de sus datos. Los gráficos se ven así (clusters en el eje x, variable a analizar en el eje y, nombres de los personajes en el hue):

## TEXTBLOB:

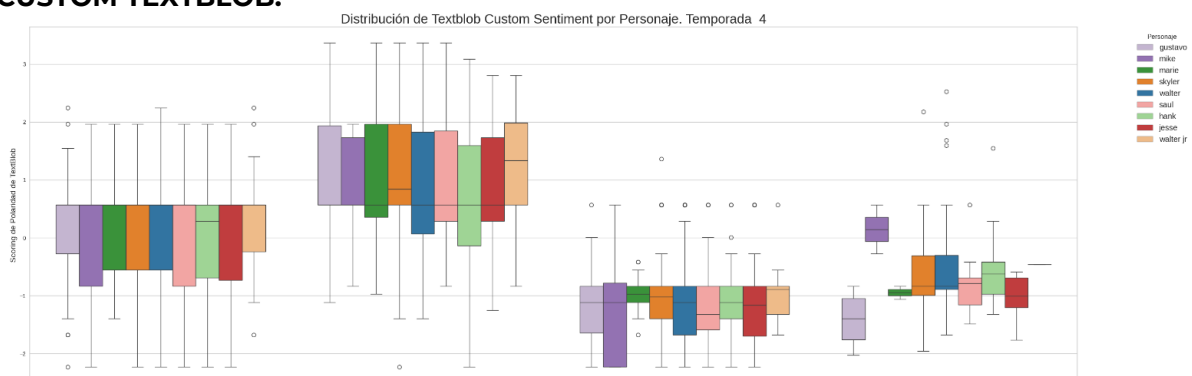
(los gráficos de esta sección son de ejemplo, ya que hay un gráfico por cada temporada, pero no los pongo todos para hacerlo más breve. En el eje x encontramos los clusters y en el eje y los valores de cada variable. **Walter** es siempre el box de color azul oscuro.)





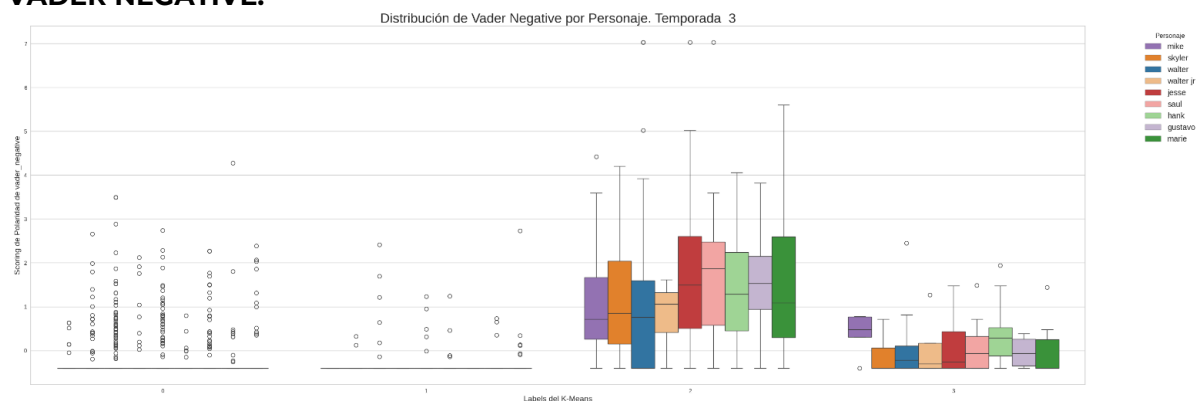
- **CLÚSTER 0:** Textblob casi inexistente. recordemos que hemos quitado el 90% de los neutros).
- **CLÚSTER 1:** Textblob de polaridad positiva.
- **CLÚSTER 2:** Textblob de polaridad negativa.
- **CLÚSTER 3:** Textblob de valores poco homogéneos que no entran en ningún otro clúster.

### CUSTOM TEXTBLOB:



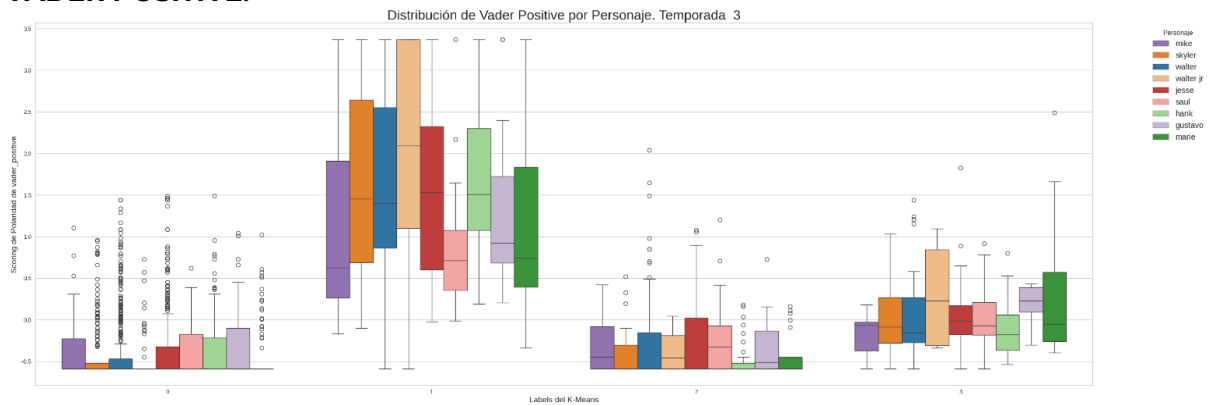
- **CLÚSTER 0:** Custom Textblob sin polaridad, neutralidad.
- **CLÚSTER 1:** Custom Textblob de polaridad positiva.
- **CLÚSTER 2:** Custom Textblob de polaridad negativa.
- **CLÚSTER 3:** Custom Textblob de polaridad no homogénea, valores inclasificables en otro clúster.

### VADER NEGATIVE:



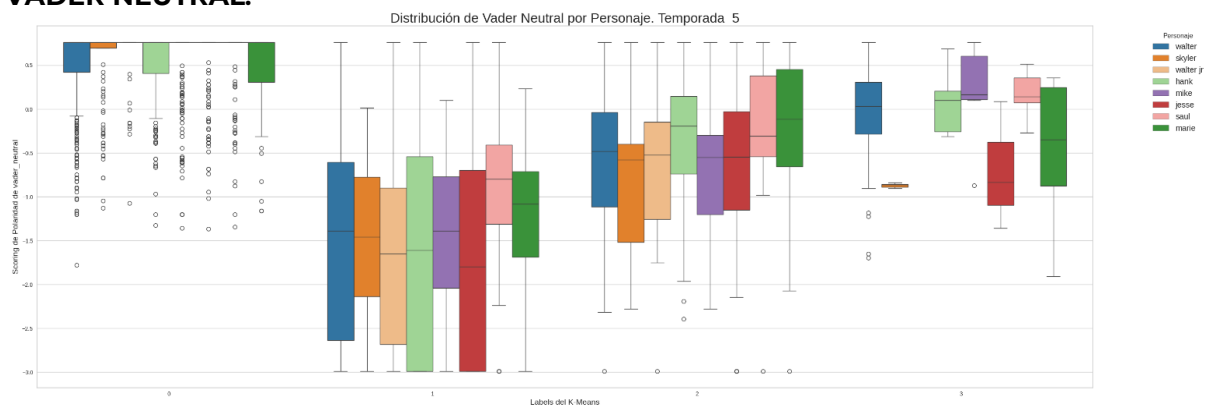
- **CLÚSTER 0:** Vader\_negative casi inexistente (quitamos el 90% de los neutros).
- **CLÚSTER 1:** Vader\_negative casi inexistente.
- **CLÚSTER 2:** Vader\_negative importante por encima de 0, **bastante** cantidad de esta variable.
- **CLÚSTER 3:** Vader\_negative bajo (entorno al cero), **baja** cantidad de esta variable.

## VADER POSITIVE:



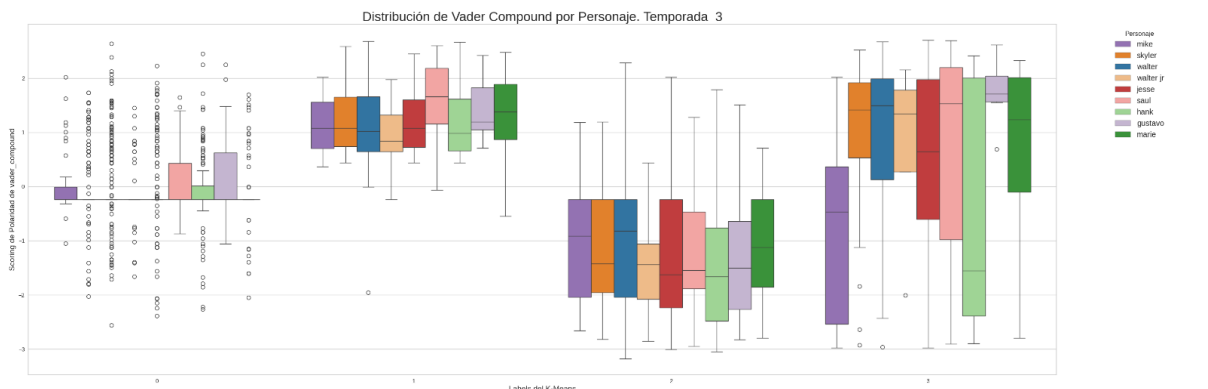
- **CLÚSTER 0:** Vader\_positive por debajo de cero o inexistente (quitamos el 90% de los neutros).
- **CLÚSTER 1:** Vader\_positive por encima del cero, variable **muy presente** en este clúster.
- **CLÚSTER 2:** Vader\_positive por debajo del cero, **baja** presencia de esta variable.
- **CLÚSTER 3:** Vader\_positive inclasificable en cualquier otro cluster.

## VADER NEUTRAL:



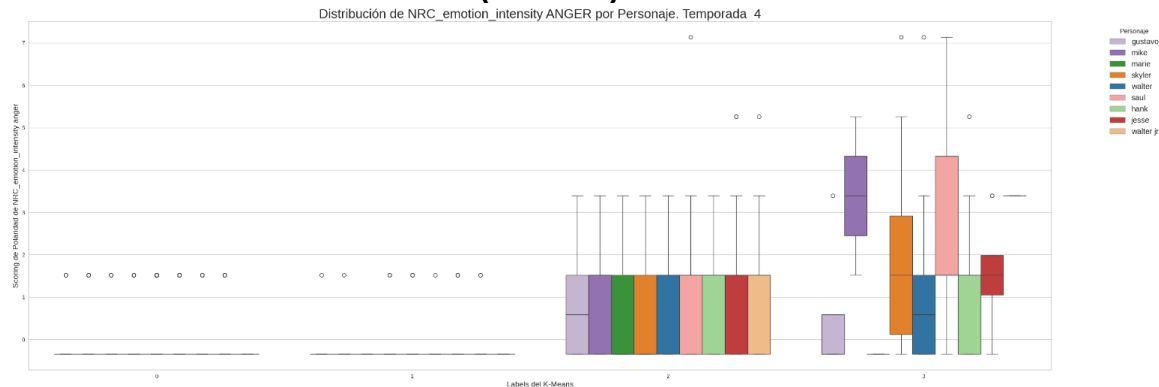
- **CLÚSTER 0:** Valores residuales de Vader\_neutral por encima del cero (quitamos el 90%)
- **CLÚSTER 1:** Vader\_neutral por debajo del cero.
- **CLÚSTER 2:** Vader\_neutral entorno al cero (neutral).
- **CLÚSTER 3:** Vader\_neutral entorno al cero pero valores residuales.

## VADER COMPOUND:



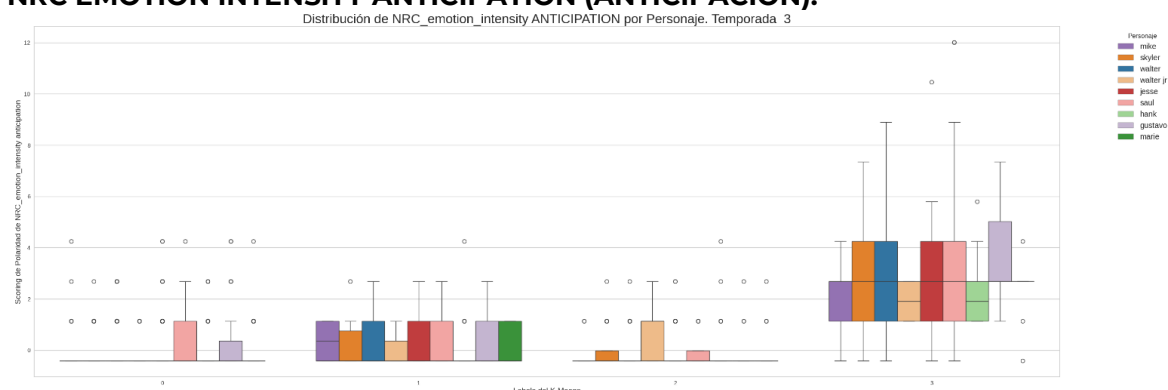
- **CLÚSTER 0:** Vader\_compound entorno a cero. Valores neutrales.
- **CLÚSTER 1:** Vader\_compound por encima de cero. Valores positivos.
- **CLÚSTER 2:** Vader\_compound por debajo de cero. Valores negativos.
- **CLÚSTER 3:** Valores extremos, raros, inclasificables.

## NRC EMOTION INTENSITY ANGER (ENFADO):



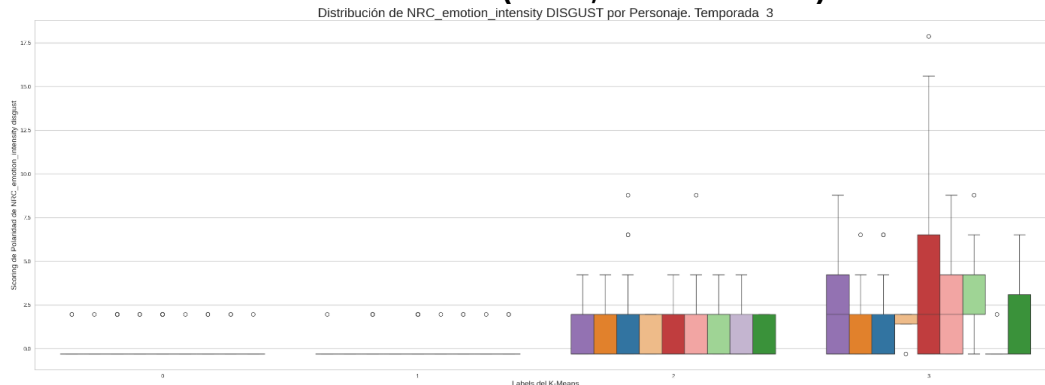
- **CLÚSTER 0:** Enfado no presente.
- **CLÚSTER 1:** Enfado no presente.
- **CLÚSTER 2:** Enfado presente pero muy homogéneo.
- **CLÚSTER 3:** Enfado presente y más extremo, aporta más información.

## NRC EMOTION INTENSITY ANTICIPATION (ANTICIPACIÓN):



- **CLÚSTER 0:** presencia no importante de anticipación.
- **CLÚSTER 1:** presencia de anticipación, pero no importante.
- **CLÚSTER 2:** presencia no importante de anticipación.
- **CLÚSTER 3:** anticipación presente y más extremo, aporta más información.

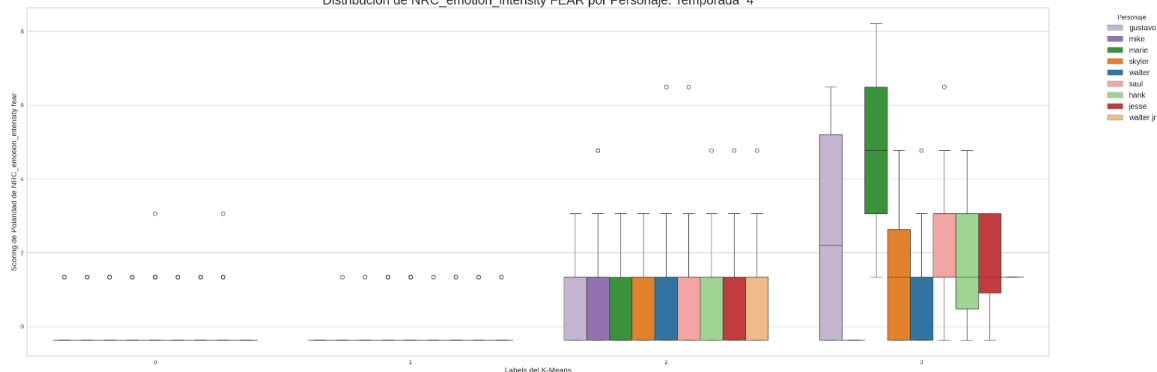
## NRC EMOTION INTENSITY DISGUST (ASCO, REPUGNANCIA):



- **CLÚSTER 0:** no hay presencia de Asco (repugnancia, disgusto).
- **CLÚSTER 1:** no hay presencia de Asco.
- **CLÚSTER 2:** presencia de Asco pero no significativa.
- **CLÚSTER 3:** presencia de Asco, valores más extremos que aportan más información.

## NRC EMOTION INTENSITY FEAR (MIEDO):

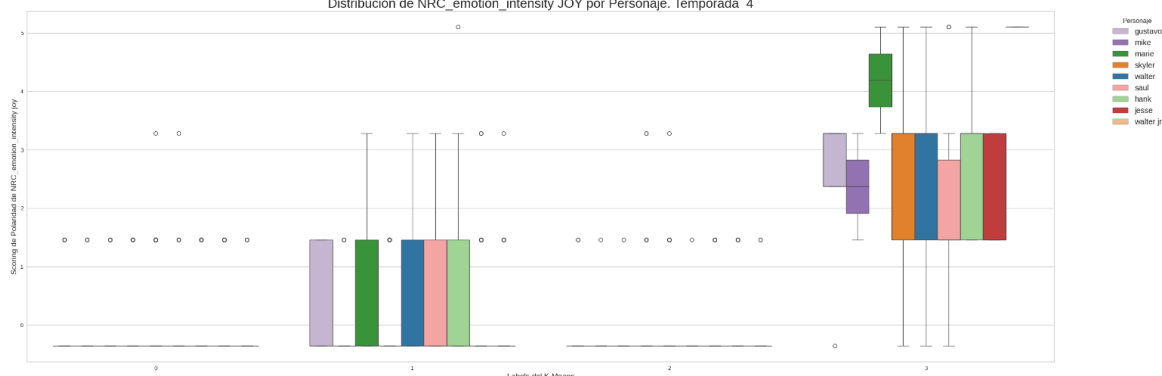
Distribución de NRC\_emotion\_intensity FEAR por Personaje. Temporada 4



- **CLÚSTER 0:** no hay presencia de Miedo.
- **CLÚSTER 1:** no hay presencia de Miedo.
- **CLÚSTER 2:** presencia de Miedo pero no significativo.
- **CLÚSTER 3:** presencia de Miedo en valores extremos, más analizable.

## NRC EMOTION INTENSITY JOY (ALEGRÍA):

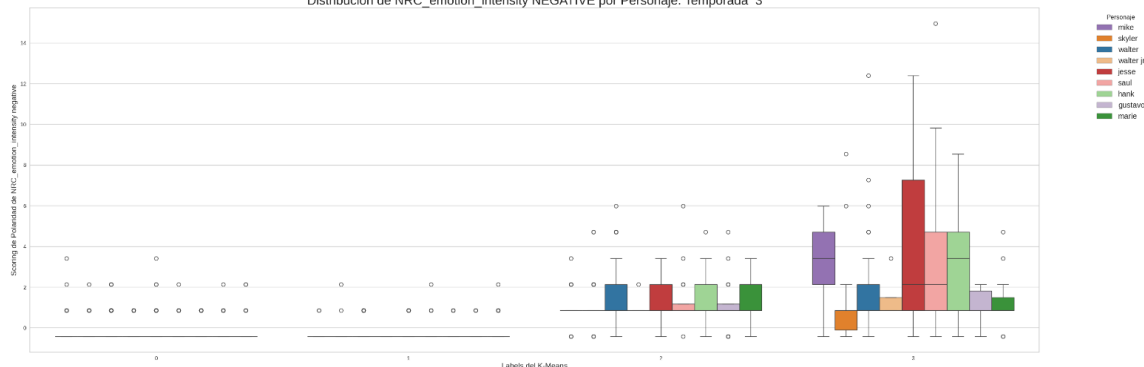
Distribución de NRC\_emotion\_intensity JOY por Personaje. Temporada 4



- **CLÚSTER 0:** no hay presencia de Alegría.
- **CLÚSTER 1:** presencia de Alegría, pero no significativa.
- **CLÚSTER 2:** no hay presencia de Alegría.
- **CLÚSTER 3:** Alegría presente en valores más altos o extremos.

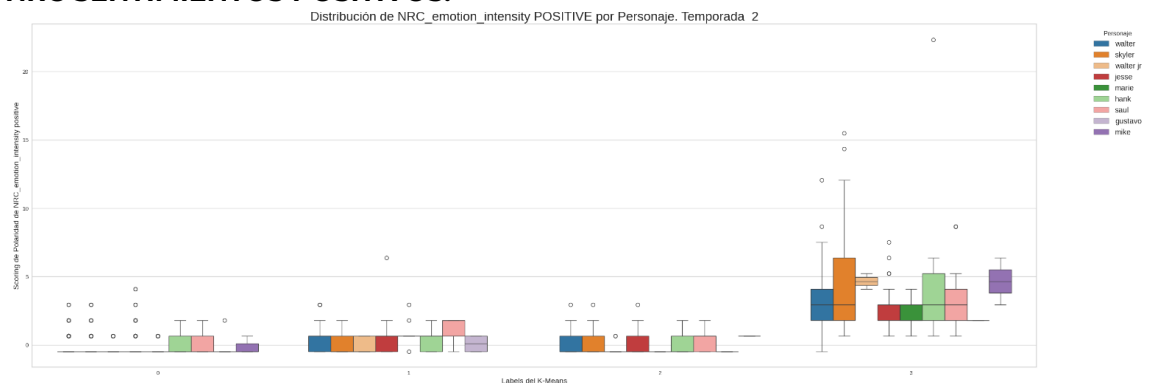
## NRC SENTIMIENTOS NEGATIVOS:

Distribución de NRC\_emotion\_intensity NEGATIVE por Personaje. Temporada 3



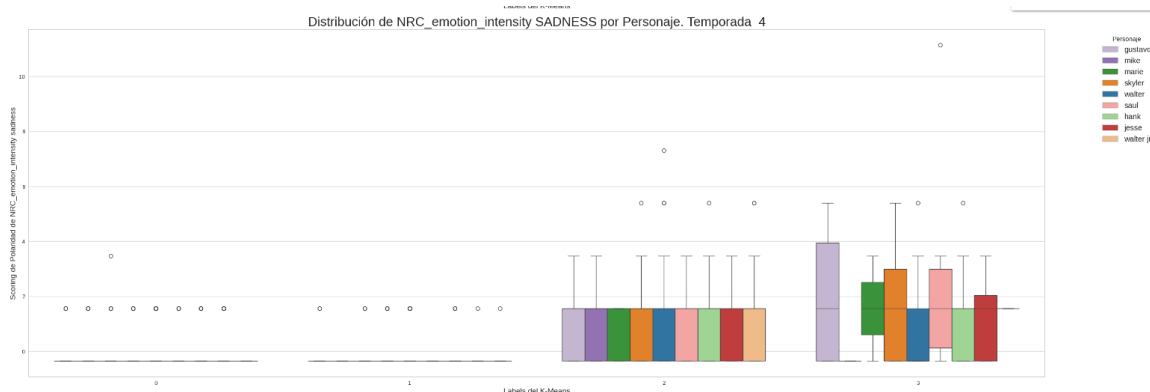
- **CLÚSTER 0:** no hay presencia de Sentimientos Negativos.
- **CLÚSTER 1:** no hay presencia de Sentimientos Negativos.
- **CLÚSTER 2:** presencia de Sentimientos Negativos, pero no significativa.
- **CLÚSTER 3:** Sentimientos Negativos presentes y analizables.

## NRC SENTIMIENTOS POSITIVOS:



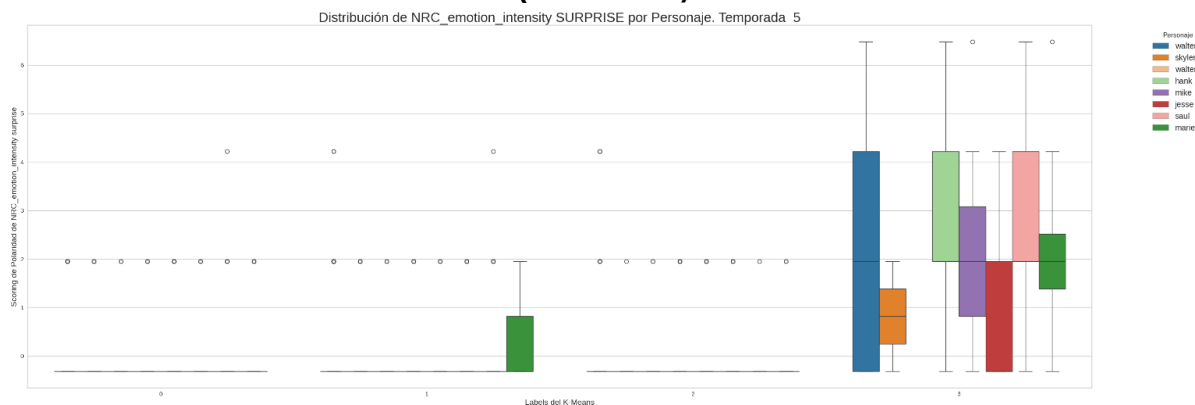
- **CLÚSTER 0:** presencia poco relevante de Sentimientos Positivos, muy matizada.
- **CLÚSTER 1:** presencia poco importante de Sentimientos Positivos.
- **CLÚSTER 2:** presencia poco importante de Sentimientos Positivos.
- **CLÚSTER 3:** Sentimientos Positivos presentes y analizables.

## NRC EMOTION INTENSITY SADNESS (TRISTEZA):



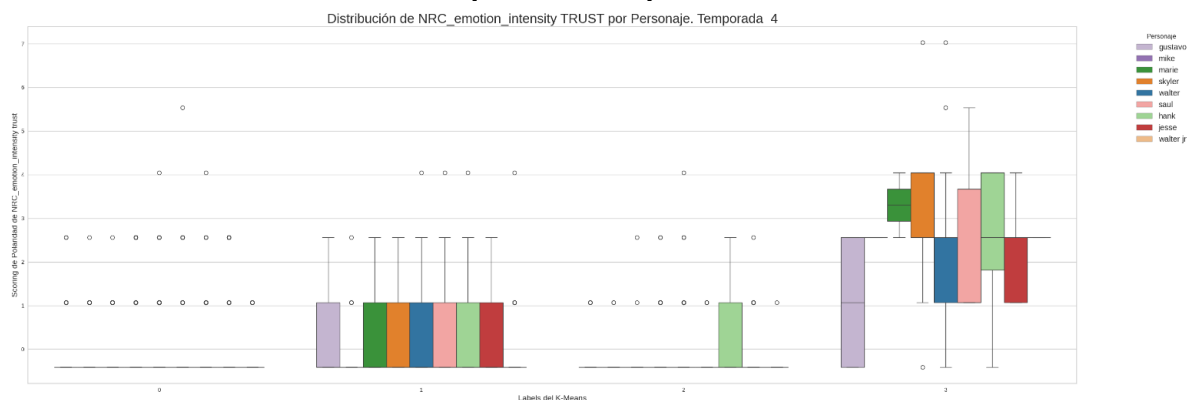
- **CLÚSTER 0:** no hay presencia de Tristeza.
- **CLÚSTER 1:** no hay presencia de Tristeza.
- **CLÚSTER 2:** presencia de Tristeza, pero no significativa.
- **CLÚSTER 3:** Tristeza presente y analizable.

## NRC EMOTION INTENSITY SURPRISE (SORPRESA):



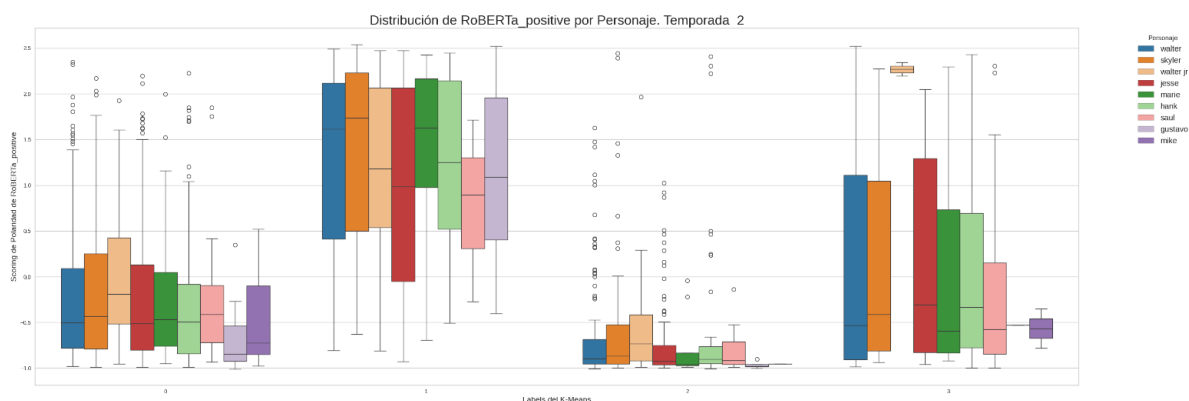
- **CLÚSTER 0:** no hay presencia de Sorpresa.
- **CLÚSTER 1:** presencia de Sorpresa, pero no significativa.
- **CLÚSTER 2:** no hay presencia de Sorpresa.
- **CLÚSTER 3:** Sorpresa presente y analizable.

## NRC EMOTION INTENSITY TRUST (CONFIANZA):



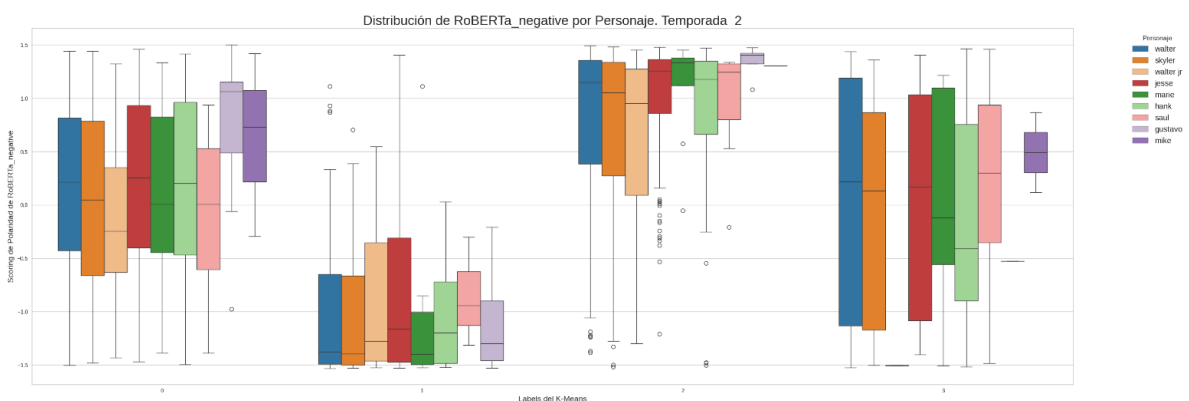
- **CLÚSTER 0:** no hay presencia de Confianza.
- **CLÚSTER 1:** presencia de Confianza pero no significativa.
- **CLÚSTER 2:** presencia de Confianza pero no significativa.
- **CLÚSTER 3:** Confianza presente y analizable.

## ROBERTA POSITIVE:



- **CLÚSTER 0:** Valores neutrales entorno al cero.
- **CLÚSTER 1:** Valores positivos por encima del cero. Es decir, **más** Roberta positive en este clúster.
- **CLÚSTER 2:** Valores negativos por debajo del cero. Es decir, **pocos** valores Roberta positive en este clúster.
- **CLÚSTER 3:** Muchos valores extremos.

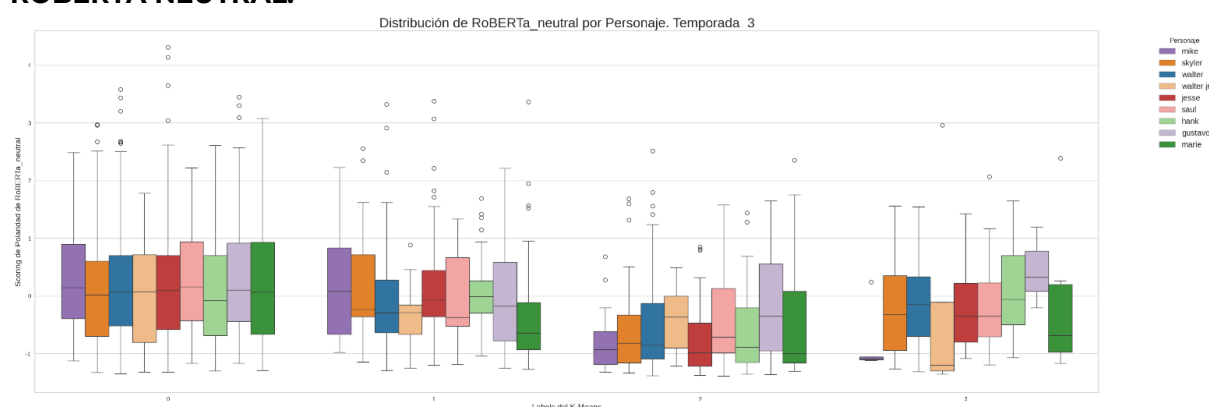
## ROBERTA NEGATIVE:





- **CLÚSTER 0:** Valores neutrales entorno al cero.
- **CLÚSTER 1:** Valores negativos por debajo del cero. Es decir, **pocos** Roberta negative en este clúster.
- **CLÚSTER 2:** Valores positivos por encima del cero. Es decir, **más** valores Roberta Negative en este clúster.
- **CLÚSTER 3:** Muchos valores extremos.

## ROBERTA NEUTRAL:



- **CLÚSTER 0:** Valores neutrales entorno al cero.
- **CLÚSTER 1:** Valores negativos por debajo del cero.
- **CLÚSTER 2:** Valores positivos por encima del cero.
- **CLÚSTER 3:** Valores extremos, entorno al cero.
- Esta variable es de todas la que más equilibrada aparece en los 4 clústeres.

## <RESUMEN DE LOS CLÚSTERS>

El **clúster 0** contiene los valores neutrales. Es tibio y gira en torno al cero sin hacer grandes aspavientos. Contiene principalmente datos provenientes de las variables Custom Textblob y RoBERTa. En este clúster podemos encontrar todo el contenido de relleno, las frases del guión sin importancia.

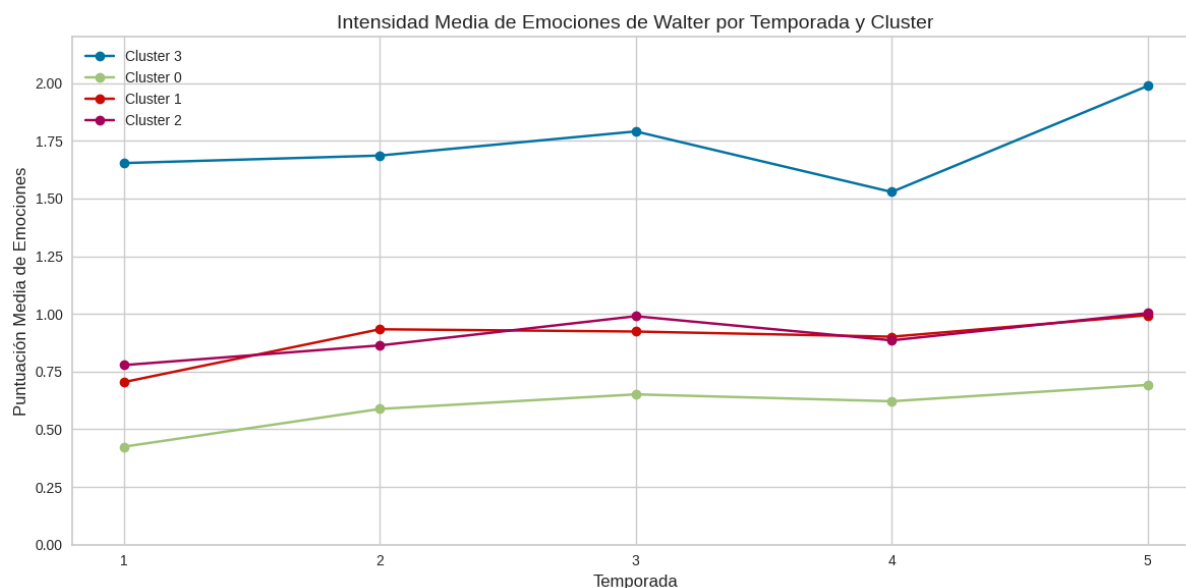
El **clúster 1**, por su parte, es el clúster del optimismo y los buenos sentimientos, pero sin caer en excesos. Contiene Textblob y Custom Textblob de polaridad positiva, mucho Vader positive, compound y neutro, pero nada de Vader negative, no hay emociones negativas del NRC, ni enfado, ni miedo, ni tristeza ni asco, sólo emociones positivas, anticipación, sorpresa, alegría, aunque, eso sí, mitigadas. También contiene RoBERTa positiva.

En cuanto al **clúster 2**, es el clúster negativo, cenizo. Contiene todo lo malo, aunque, al igual que en el caso anterior, sin caer en extremos. Contiene Textblob y Custom Textblob de polaridad negativa, casi todo el Vader negativo y prácticamente nada del positivo, aunque también una parte importante del compound y del neutro, todas las emociones negativas del NRC aunque suavitas. No tiene nada de RoBERTa de polaridad positiva pero sí bastante de RoBERTa de polaridad negativa.

Por último, el **clúster 3** parece captar escenas o diálogos en los que el sentimiento se vuelve especialmente intenso. Es probable que el K-means haya agrupado estas interacciones o

diálogos porque comparten ciertas características (quizá en el lenguaje o el tono) que los diferencian de los demás grupos. La gran variabilidad en las puntuaciones podría indicar que el clúster 3 **representa los diálogos más cargados emocionalmente o polarizados**. Esto significa que la mayoría de los personajes de este grupo podrían estar experimentando altibajos emocionales más intensos en comparación con otros grupos, lo que daría lugar a valores más extremos.

En concreto, para Walter a lo largo de las 5 temporadas:

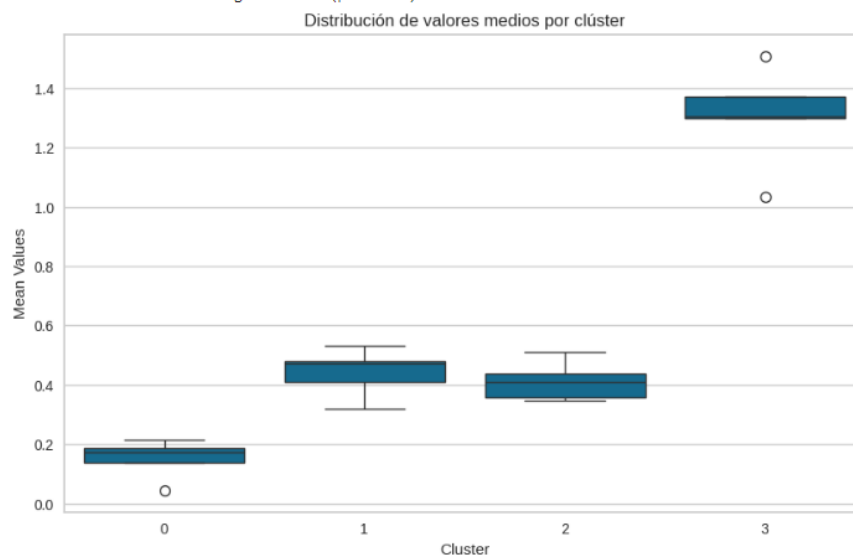


Seguimos con nuestro análisis de clustering. El siguiente paso es comparar los clústers entre ellos para encontrar grupos estadísticamente significativos. Es decir, ver si cada uno de los clústers es significativamente diferente de los demás. Para ello vamos a realizar el **Kruskal-Wallis** test. La selección de esta prueba no-paramétrica responde a nuestra incertidumbre sobre la forma de la distribución de los datos. Ya que Kruskal-Wallis funciona bien con datos que no siguen una distribución normal o que presentan un gran número de outliers, nos parece la mejor opción.

Lo que esperamos encontrar con esta prueba son p-values menores de 0,05 (es decir, la probabilidad de observar los resultados del estudio, u otros más alejados de la hipótesis nula, si la hipótesis nula fuera cierta [...]). El valor p ayuda a diferenciar resultados que son producto del azar del muestreo, de resultados que son estadísticamente significativos. *Valor-p. Wikipedia*).

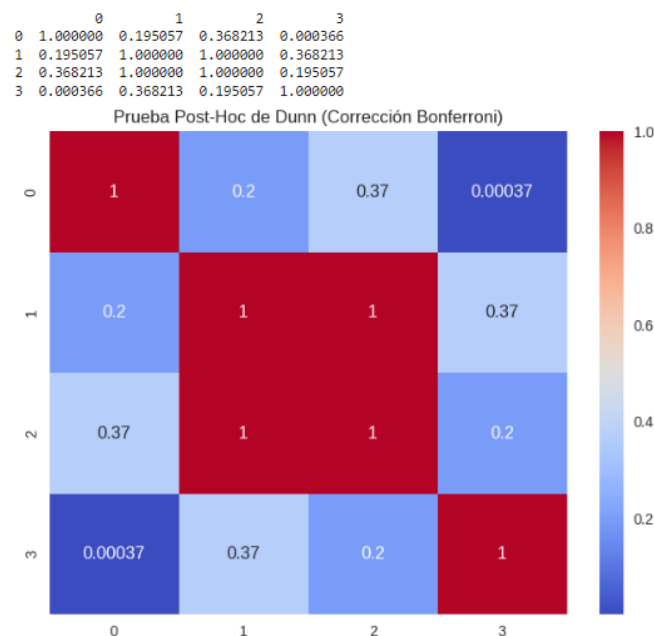
Vemos que el p-value es menor de 0,05 en al menos uno de los clústers:

Resultado estadísticamente significativo ( $p=0.00106$ ). Al menos un clúster es diferente.



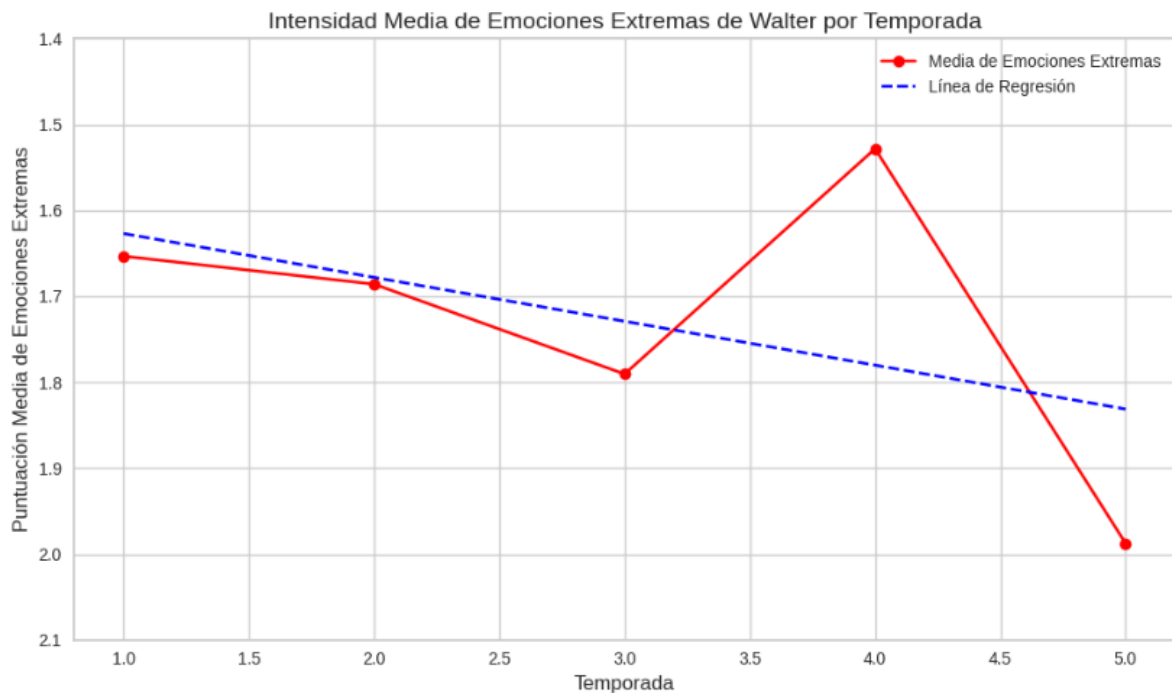
Continuamos nuestro análisis y realizamos la **prueba Posthoc de Dunn** para verificar cuáles de los grupos son distintos. Esta prueba utiliza los mismos datos que Kruskal-Wallis para probar las diferencias entre dos grupos cualesquiera. En concreto, la aproximación de la prueba z de Dunn (1964) se calcula como la diferencia en las puntuaciones medias de rango dividida por la estimación de la varianza agrupada de rango para dos grupos.

Vemos que el grupo con diferencias significativas ( $p\text{-value} < 0,05$ ) lo componen los clústers 0 y 3, que recordemos que eran el de las emociones neutras y el de las emociones extremas respectivamente:



Si tomamos en cuenta este resultado y nos centramos sólo en el clúster 3 de las emociones extremas que es estadísticamente significativo, podemos observar el ascenso, o invirtiendo el eje y el descenso, de Walter a la maldad:

```
Temporada
1    1.653396
2    1.685942
3    1.790563
4    1.528337
5    1.987540
dtype: float64
```



## 5. CONCLUSIONES Y TRABAJO FUTURO\_

En este trabajo hemos analizado la caída moral de Walter White, protagonista de la serie *Breaking Bad* desde la perspectiva del NLP (Procesamiento de Lenguaje Natural) y el Análisis de Sentimientos. Hemos empleado técnicas basadas en reglas, lexicones, machine learning supervisado (Naïve Bayes) y no-supervisado (Clustering con K-Means).

Una de las preguntas más recurrentes al realizar este tipo de análisis es, ¿por qué fallan algunos modelos? Lo cierto es que, a lo largo de la serie, Walter experimenta conflictos internos que a veces revelan su humanidad. Hay momentos en los que muestra preocupación por su familia y se enfrenta a la culpa por las consecuencias de sus acciones. Estos momentos pueden influir en el análisis de sentimientos, ya que un modelo podría interpretar la complejidad emocional de Walter como menos negativa en ciertas escenas, incluso si su comportamiento sigue siendo perjudicial.

Además, *Breaking Bad* es conocida por su ambigüedad moral. A medida que Walter se adentra más en el mundo criminal, las acciones de los personajes, incluidos los antagonistas, son complejas y matizadas. Este enfoque puede dificultar que un modelo de análisis de sentimientos capture las sutilezas de los sentimientos del personaje.

No obstante, la evolución de Walter es significativa porque sus motivaciones cambian. Al

principio, su acción puede parecer comprensible, casi altruista. Pero conforme avanza la historia, las acciones de Walter se vuelven cada vez más egoístas y destructivas al tiempo que se siente cada vez más fuerte e invencible. En este contexto, un análisis de sentimientos podría mostrar un descenso en la negatividad y un aumento en la positividad incluso cuando sus acciones se vuelven más cuestionables.

Además, si Walter justifica una acción violenta o una mentira con la idea de que lo hace por su familia, el análisis de sentimientos podría no captar la verdadera gravedad de su comportamiento, reflejando en cambio su perspectiva racionalizada y menos negativa de lo que es en realidad.

Asimismo, las herramientas de análisis de sentimientos exploradas en este trabajo tienen, más o menos todas, dificultades para captar el contexto, la ironía, el sarcasmo, los dobles sentidos y los matices de la comunicación oral. Un estudio en profundidad de una muestra de frases y las puntuaciones que les atribuyen cada uno de los modelos explorados, nos hacen ver que los lexicones son herramientas con grandes lagunas para llevar a cabo con éxito este tipo de tarea.

Además de todo esto, hemos de reconocer que nuestro trabajo no se ha desarrollado en condiciones óptimas. El dataset inicial no está completo. La falta de identificación de los personajes de varios capítulos de la Temporada 5 ha podido influir negativamente en la claridad de los resultados obtenidos. Asimismo, el hecho de haber extraído manualmente los personajes de la temporada 4, hace que esta temporada esté compuesta por frases más cortas y separadas que en las tres temporadas precedentes. El formato de las frases debería haberse homogeneizado antes de realizar el trabajo.

En conclusión:



#### > DESHUMANIZACIÓN

Hemos conseguido modelar la deshumanización de Walter White a medida que transcurren los episodios mediante el análisis de los datos.



#### > EXTREMISMO

También hemos conseguido modelar el aumento de picos de sentimientos y emociones extremas de Walter a lo largo de las temporadas.



#### > MALDAD

Tanto la deshumanización y como aumento de la frecuencia de emociones extremas prueban la modificación psicológica sufrida por este personaje, conduciéndole hasta la maldad pura y simple.

#### >>>> TRABAJOS FUTUROS/

- Completar el dataset inicial con los datos faltantes y homogeneizar las frases. Hacer que las frases sean más cortas para que los lexicones las puedan analizar mejor.
- Analizar el descenso a la maldad de otros personajes: Jesse, Skyler, etc. y compararlos con Walter.

- Realizar Fine-Tuning del modelo de Deep Learning RoBERTa para obtener resultados más precisos, ya que puede capturar mejor que los léxicos los matices y las sutilezas emocionales.
- Realizar un análisis temporal y dinámico de los cambios emocionales de Walter para identificar los momentos clave en los que cambia de comportamiento.
- Analizar la co-ocurrencia de emociones: ver qué emociones ocurren “juntas”.
- Analizar imágenes del rostro de Walter para captar cambios físicos y gestuales que puedan darnos más información sobre su descenso a la maldad.

## 6. REFERENCIAS BIBLIOGRÁFICAS

1. Forever Dreaming. (n.d.). **Breaking Bad - Episode Transcripts**. Forever Dreaming. <https://transcripts.foreverdreaming.org/viewforum.php?f=165>
2. Genius. (n.d.). **Breaking Bad: Cornered [Script]**. Genius. <https://genius.com/Breaking-bad-cornered-script-annotated>
3. Rietvelt, J. (2019). *Sentiment analysis: A comparison of methods for the classification of the sentiment of tweets* (Bachelor's thesis). University of Twente. [https://essay.utwente.nl/78791/1/Rietvelt\\_BA\\_EEMCS.pdf](https://essay.utwente.nl/78791/1/Rietvelt_BA_EEMCS.pdf)
4. Datumbox. (n.d.). *The importance of neutral class in sentiment analysis*. <https://blog.datumbox.com/the-importance-of-neutral-class-in-sentiment-analysis/>
5. López, F. (2021). Emociones y creencias: Un análisis desde la filosofía. *Isegoria*, 64, 25-47. <https://isegoria.revistas.csic.es/index.php/isegoria/article/view/582/583>
6. Fesenko, P. (2021, October 11). Best open-source models for sentiment analysis: Part 1 — Dictionary models. *Medium*. <https://medium.com/@pavlo.fesenko/best-open-source-models-for-sentiment-analysis-part-1-dictionary-models-ece79e617653>
7. Davidson, T. (n.d.). *Hate speech lexicons*. GitHub. <https://github.com/t-davidson/hate-speech-and-offensive-language/tree/master/lexicons>
8. NLP Progress. (n.d.). *NLP progress*. <https://nlpprogress.com/>
9. Dey, D. (2022). A survey on deep learning for natural language processing: The state of the art. *Multimedia Tools and Applications*, 81(8), 11517–11539. <https://link.springer.com/article/10.1007/s11042-022-13428-4>
10. Zubair, H. (2022). The state of the art of natural language processing. *Dint*, 5(3), 707–720. <https://direct.mit.edu/dint/article/5/3/707/115133/The-State-of-the-Art->



## of-Natural-Language

11. Thematic. (n.d.). *What is sentiment analysis?*  
<https://getthematic.com/sentiment-analysis/#:~:text=Sentiment%20analysis%20uses%20machine%20learning,base d%20and%20automated%20sentiment%20analysis>
12. Loria, S. (2018). *TextBlob: Simplified text processing*.  
<https://textblob.readthedocs.io/en/dev/>
13. PyPI. (n.d.). *Profanity-check*. <https://pypi.org/project/profanity-check/>
14. Hex Technologies. (n.d.). *VADER sentiment analysis*. <https://hex.tech/use-cases/sentiment-analysis/vader-sentiment-analysis/>
15. SentiWordNet. (n.d.). GitHub. <https://github.com/aesuli/SentiWordNet>
16. Mohammad, S. (n.d.). *NRC emotion lexicon*.  
<https://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm>
17. Adictos al Trabajo. (2023, July 27). *NLTK en Python: Introducción al procesamiento de lenguaje natural*.  
<https://adictosaltrabajo.com/2023/07/27/nltk-python/>
18. Bebow. (n.d.). *Segmentación avanzada con algoritmo K-means: Ejemplo*.  
<https://bebow.es/segmentacion-avanzada-con-algoritmo-k-means-ejemplo/>
19. Hugging Face. (n.d.). *Roberta*.  
[https://huggingface.co/docs/transformers/model\\_doc/roberta](https://huggingface.co/docs/transformers/model_doc/roberta)
20. Universidad de Oviedo. (n.d.). *K-means clustering*.  
[https://www.unioviedo.es/compnum/laboratorios\\_py/kmeans/kmeans.html](https://www.unioviedo.es/compnum/laboratorios_py/kmeans/kmeans.html)
21. UOC. (n.d.). *Data mining: Módulo 5: Agregación (clustering)*.  
[https://openaccess.uoc.edu/bitstream/10609/138187/24/Data%20mining\\_M%C3%B3dulo%205\\_Agregaci%C3%B3n%20%28clustering%29.pdf](https://openaccess.uoc.edu/bitstream/10609/138187/24/Data%20mining_M%C3%B3dulo%205_Agregaci%C3%B3n%20%28clustering%29.pdf)
22. Wikipedia contributors. (n.d.). Valor p. *Wikipedia, la enciclopedia libre*.  
[https://es.wikipedia.org/wiki/Valor\\_p](https://es.wikipedia.org/wiki/Valor_p)
23. DATAtab Team. (2024). *Prueba de Kruskal-Wallis*.  
<https://datatab.es/tutorial/kruskal-wallis-test>
24. Amat Rodrigo, J. (2016, enero). *Test Kruskal-Wallis. Ciencia de Datos*.  
[https://cienciadedatos.net/documentos/20\\_kruskal-wallis\\_test](https://cienciadedatos.net/documentos/20_kruskal-wallis_test)
25. Rajesh, A. (n.d.). *A Post-hoc Test for Kruskal-Wallis. The Analysis Factor*.  
<https://www.theanalysisfactor.com/dunns-test-post-hoc-test-after-kruskal-wallis/>
26. *Dunn-Bonferroni post-hoc test*. (s. f.). Easy Made Stat.  
<https://help.easymedstat.com/support/solutions/articles/77000536997-dunn-bonferroni-post-hoc-test>